Polk

Die besten Tips & Tricks



DATA BECKER

Andreas Polk

Commodore 64
Die besten Tips & Tricks

DATA BECKER

1. Auflage 1988

ISBN 3-89011-281-1

DATA BECKER GmbH
Merowingerstr. 30
4000 Düsseldorf

Text verarbeitet mit Word 4.0, Microsoft
Ausgedruckt mit Hewlett Packard LaserJet II
Druck und Verarbeitung Elsner-Druck, Berlin

Wichtiger Hinweis:
Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technischen Angaben und Programme in diesem Buch wurden von dem Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

Vorwort

C64 Tips und Tricks. Warum noch ein Buch, wo es doch schon genug Bücher dieser Art auf dem Markt gibt? Diese Frage ist natürlich berechtigt. Doch dieses Buch unterscheidet sich etwas von den bisher erschienenen Exemplaren: Das Ziel dieses Buches ist es nicht, Ihnen seitenlange Maschinensprachelistings 'vorzulegen', womit dann ein toller Effekt erreicht wird.

Dieses Buch richtet sich an diejenigen, die BASIC in den Grundzügen gelernt haben und nun etwas mehr wissen möchten. Dabei ist dieses Buch kein Lehrbuch. Ich möchte Ihnen hier vielmehr kurze, aber effektive Tips und Tricks geben, mit denen Sie arbeiten können. Dabei werden die Tips nicht nur mitgeteilt, sondern es wird auch meistens das 'Warum?' erklärt.

Im Grafikteil möchte ich Ihnen dann eine kurze Einführung in die Grafikprogrammierung des C64 geben, denn gerade hier entsteht oft eine Hemmschwelle, da das BASICV2 die Grafikprogrammierung in keiner Weise unterstützt.

Das letzte Kapitel handelt über das Thema GEOS, das neue Betriebssystem, das jedem C64 beigelegt wird. In diesem Kapitel wird jedoch keine Einführung im Sinne einer Erklärung der Menüpunkte gegeben, sondern es zeigt Ihnen, was sich mit GEOS alles bewerkstelligen läßt.

Zum Schluß möchte ich noch einige Danksagungen 'loswerden'. Zuerst möchte ich "Hasi" Dirk von dem Bussche danken, der mir bei der Fertigstellung dieses Buches half. Insbesondere war er an der Fertigstellung der Kapitel 4 und 8 beteiligt. Weiter möchte ich Herrn Klaus Petring danken, der mir bei Kapitel 10 behilflich war.

Ein spezieller Dank sei hier noch an Holger Breit gerichtet, der immer bereit war, dieses Buch Korrektur zu lesen und mir mit seinen Verbesserungsvorschlägen geholfen hat, das Buch optimal

zu gestalten. Des weiteren sei noch allen gedankt, die Rücksicht darauf nahmen, daß ich manchmal für mehrere Tage nicht aus meinem Zimmer herauskam, außer um Nahrung zu mir zu nehmen. Doch nun genug der Vorrede. Viel Spaß beim Lesen!

Andreas Polk   Düsseldorf, Juni 88

Zum Umgang mit diesem Buch

Die Beispiele in diesem Buch sind alle ausführlich erklärt. Diesbezüglich dürfte es also kaum zu Ungereimtheiten kommen. Um Tastenkombinationen von Text zu unterscheiden, habe ich die Bezeichnung der Taste in Klammern gesetzt. So bedeutet z.B.

```
PRINT "<CRL/SCREEN>"
```

daß nach den Anführungsstrichen die Tastenkombination gedrückt werden soll, wodurch das Löschen des Bildschirms erreicht wird. Schreibe ich jedoch

```
PRINT "Führen Sie bitte einen CRL/SCREEN aus!"
```

so sollten Sie hier den gesamten Text eingeben. Steht eine Tastenkombination also in Klammern, so heißt das, daß die entsprechenden Tasten gedrückt werden sollen. Des weiteren sind alle BASIC-Befehlswörter groß geschrieben. Dadurch wird eine größere Übersichtlichkeit erreicht. Das Buch setzt voraus, daß man sich mit Bitoperationen auskennt. Sollte dies noch nicht der Fall sein, so können Sie sich im Anhang unter dem Stichwort Bitoperationen die entsprechenden Grundlagen aneignen.

Inhaltsverzeichnis

1. Die Datasette

Die Datasette ist das wohl preiswerteste Speichergerät für den C64. Obwohl die Floppy bei den meisten Anwendern schon die Datasette vertrieben hat, möchte ich Ihnen dennoch ein paar Datasetten-Tips und -Tricks verraten, die Sie dann recht einfach auch in Ihre Programme einbauen können.

1.1 Steuern der Datasette von BASIC aus

Vielleicht ist Ihnen schon einmal die lästige Eigenschaft der Datasette aufgefallen, immer nach einem Lade- oder Speichervorgang selbsttätig zu stoppen. Um nun wieder ein Programm zu laden, muß immer erst die STOP-Taste und dann die PLAY-Taste gedrückt werden. Was stoppt denn eigentlich den Motor der Datasette? Nun, nichts anderes als der Computer. Das können wir nun ausnützen und die Datasette vom Computer aus steuern! Um dies zu demonstrieren, drücken Sie bitte einmal die PLAY-Taste, und geben Sie im Direktmodus bitte folgende zwei Befehle ein:

```
POKE 192,1
POKE 1,PEEK (1) OR 32
```

Und siehe da, die Datasette stoppt, als hätten Sie von Hand die STOP-Taste betätigt. Nun, wie funktioniert der Trick: Soll der Motor der Datasette eingeschaltet werden, so muß in der Speicherstelle 192 ein Wert ungleich null stehen. Deshalb wird der Wert Eins in diese Speicherstelle geschrieben. Weiterhin muß das 5. Bit der Speicherstelle 1 gesetzt werden, was mit dem Befehl POKE 1,PEEK (1) OR 32 geschieht. Ist in der Speicherstelle das Bit 5 gesetzt, so wird der Motor der Datasette abgeschaltet. Es ist aber genauso möglich, den Motor starten zu lassen. Geben Sie dazu bitte folgenden Befehl ein:

```
POKE 192,0
```

Es wird hier wieder der Wert 0 in die Speicherstelle 192 geschrieben. Mit den folgenden Befehlen läßt sich abfragen, ob eine Taste der Datasette gedrückt wurde oder nicht:

```
IF PEEK (1) = 55 THEN PRINT "KEINE TASTE GEDRÜCKT"
IF PEEK (1) =  7 THEN PRINT "TASTE GEDRÜCKT"
```

Wollen Sie in einem Programm solange warten, bis der Benutzer die STOP-Taste gedrückt hat, dann können Sie dies durch den Befehl

```
WAIT 1,16
```

erfahren. Dadurch wird der Computer nämlich veranlaßt, solange zu warten, bis in Speicherzelle 1 das 4. Bit gesetzt ist, was nur der Fall ist, wenn die STOP-Taste der Datasette gedrückt wurde. Im folgenden habe ich Ihnen noch einmal alle Möglichkeiten zum Steuern der Datasette aufgelistet:

```
POKE 192,1                 (Motor aus)
POKE 1,PEEK (1) OR 32

POKE 192,0                 (Motor an)
POKE 1,PEEK (1) AND 39

WAIT 1,16                  (Warten auf STOP-Taste)
IF PEEK (1) = 55 THEN ...  (Warten auf PLAY-Taste)
```

1.2 Ein Kopierschutz für die Datasette

Wer hat nicht schon einmal ein Programm geschrieben, das er vor unerlaubtem Kopieren sichern wollte? Mit der Datasette ist es sehr einfach, einen Kopierschutz zu programmieren. Wird ein Programm auf Datasette abgespeichert, so kann man einen Programmnamen mit einer Länge von bis zu 172 Buchstaben Länge angeben. Beim Laden dieses Programmes werden lediglich 16 Zeichen höchstens angegeben.

'Doch was nützt uns das jetzt?' werden Sie sich sicher fragen. Nun, beim Laden eines Programmes werden die restlichen 160 Buchstaben des Namens in den sogenannten Kassettenpuffer geladen. Kopiert nun jemand Ihr Programm, ohne daß Sie es wollen, so kennt er natürlich nicht den Rest des Namens. Beim Abspeichern gibt er deshalb nur einen höchstens 16 Zeichen langen Namen ein, den woher soll er ahnen, daß das Programm einen 17 Zeichen langen Namen hat? Das können Sie sich zu Nutzen machen.

Ich habe hier nun für Sie ein kleines Programm geschrieben, das das 17. Zeichen des Namens abfragt:

```
100 IF PEEK (849) = ASC("A") THEN SYS64738
110 PRINT "ES HANDELT SICH HIER UM EIN ORIGINALPROGRAMM!"
120 PRINT "VIEL SPAß BEI DESSEN BENUTZUNG!"
```

Ab der Speicherstelle 849 befinden sich 160 Bytes, in denen der restliche Name des Programmes steht. Dieser Teil des Namens läßt sich durch den PEEK-Befehl auslesen. Voraussetzung dafür ist jedoch, daß der Programmname 16 Zeichen lang ist, zusätzlich der restlichen Zeichen. Ein Beispiel für solch einen Namen ist:

```
1234567890123456A
```

Die ersten 16 Zeichen werden beim Ladevorgang angezeigt, das siebzehnte jedoch nicht. Es steht im Speicher an der Adresse 849. Durch die Befehlsreihe in Zeile 100 wird ein RESET ausgelöst, falls in der Speicherstelle 849 ein anderer Wert als der Code für das Zeichen A steht. Ansonsten fährt das Programm mit Zeile 110 fort. Sie sollten diese Zeilen immer vor den Programmanfang stellen. Sie können statt des Codes für das Zeichen A natürlich auch einen anderen Code abfragen. Dafür müssen Sie nur Zeile 100 ändern.

1.3 Beschleunigung des Ladevorgangs

Programme, die auf Datasette gespeichert werden, sind - um eventuellen Ladefehlern vorzubeugen - doppelt abgespeichert. Es ist jedoch Unsinn, ein Programm zweimal zu laden, da dies fast doppelt so lange dauert. Die Ladezeit läßt sich durch folgende Programmzeilen, die Sie an den Anfang des zu ladenden Programmes stellen sollten, verkürzen.

```
POKE 45,PEEK (831)
POKE 46,PEEK (832)
CLR
```

Sie können nun nach dem ersten Ladevorgang die STOP-Taste drücken, damit das Programm nicht unnützerweise zweimal geladen wird. Durch die drei Zeilen wird bewirkt, daß die Zeiger auf das BASIC-Ende richtig gesetz, und dann alle Variablen gelöscht werden. Würde dies nicht gemacht, so könnte es leicht vorkommen, daß während des Programmablaufes einfach Fehler auftreten, die sonst nicht vorhanden wären.

Eine zweite Möglichkeit, den Ladevorgang zu beschleunigen, ist das Abbrechen der Wartepause, nachdem der Computer die Meldung ausgegeben hat, daß er das Programm gefunden hat. Drücken Sie dazu einfach entweder die CRTL-, die SPACE- oder die COMMODORE-Taste. Ein Drücken des Feuerknopfes des Joysticks, der sich in Port 1 befindet, bewirkt das gleiche.

Manche Maschinenprogrammierer legen kleine Programm-Routinen im Kassettenpuffer ab. Auch wenn man viel Speicherplatz für Sprite-Daten braucht, ist es sinnvoll, diese im Kassettenpuffer abzulegen. Viele Leute sind der Meinung, daß es dann nicht mehr möglich sei, Programme von der Datasette zu laden, da sonst die Maschinensprache-Routine überschrieben würde. Daß es dennoch möglich ist, von der Datasette zu laden, ohne Programme, die im Kassettenpuffer abgelegt sind, zu überschreiben, zeigen folgende zwei Befehle, die im Direktmodus eingegeben werden müssen:

```
POKE 178,0
POKE 179,4
```

Diese beiden Pokes bewirken, daß die Zeiger, die auf den Anfang des Kassettenpuffers zeigen, nun 'umgebogen' werden und auf die Bildschirmspeicher zeigen. Dadurch werden die Daten, die sich im Kassettenpuffer befinden, nicht zerstört. Sie können natürlich auch andere Speicherbereiche als den Bildschirmspeicher angeben. Dazu müssen Sie nur die Parameter der POKE-Befehle ändern (siehe Anhang).

1.4 Retten eines Programmes nach einem LOAD-ERROR

Die Datasette ist dafür bekannt, daß sie ein unsicheres Speichergerät ist. Es kommt manchmal vor, daß ein schlichter LOAD-ERROR nach einem versuchten Ladevorgang auftaucht. Doch noch müssen Sie nicht verzagen.

Die Datasette speichert - wie in Kapitel 1.3 schon erwähnt - jedes Programm doppelt ab. Beim Laden wird dann der erste Teil ganz normal geladen und mit dem zweiten Teil verglichen. Stimmen manche Teile nicht überein, so kommt es zu dem gefürchteten LOAD-ERROR.

In Wirklichkeit ist der LOAD-ERROR also gar kein richtiger Ladefehler, sondern es sind lediglich die beiden Programmteile unterschiedlich. Sollte bei Ihnen ein LOAD-ERROR auftreten, so versuchen Sie, das Programm zu listen. Wenn dies noch einigermaßen funktioniert, so ist Ihr Programm nicht verloren.

Sie müssen nur noch den Zeiger auf das BASIC-Ende richtig setzen, da die Datasette diese dem Computer erst übergibt, wenn der Ladevorgang funktioniert hat, was bei einem LOAD-ERROR nicht der Fall ist.

Die Werte für den Zeiger auf das BASIC-Ende finden Sie im Kassettenpuffer, und zwar bei Adresse 831/832. Diese Werte müssen dem Computer lediglich mitgeteilt werden, was durch folgende Befehle geschieht:

```
POKE 46,PEEK (832)
POKE 47,PEEK (831)
POKE 48,PEEK (832)
POKE 49,PEEK (831)
POKE 50,PEEK (832)
```

Nun läßt sich Ihr Programm wie gewöhnlich durch RUN starten und durch SAVE"PRG.Name",1 abspeichern. Da auch dieser Trick nicht immer funktioniert, ist das sicherste Mittel gegen defekte Datenträger immer noch das Anfertigen einer Sicherheitskopie!

2. Tips und Tricks zur Speicherbelegung

Der Speicher des C64 ist für spezielle Aufgaben aufgeteilt. So beginnt z.B. der Bildschirmspeicher ab Adresse 1024, der Kassettenpuffer ab 828 usw. Manchmal wäre es wünschenswert, wenn der kassettenpuffer in einem anderen Speicherbereich läge, um in seinem ursprünglichen Bereich z.B. Sprite-Daten, kleine Maschinenprogramme oder ähnliche Daten abzulegen. Zum Glück haben wir die Möglichkeit, die Speicheraufteilung teilweise selbst zu bestimmen, und genau darüber handelt dieses Kapitel.

2.1 Verschieben des Kassettenpuffers

Der Kassettenpuffer liegt nach dem Einschalten bei der Adresse 828 und ist 190 Bytes lang. Er reicht also bis zu Adresse 1019. Dieser Bereich wird nur genutzt, wenn etwas auf die Datasette geschrieben oder von ihr gelesen wird. Sonst ist er unbenutzt. Deshalb werden in diesen Speicherbereich gerne kleine Maschinenprogramme oder die Sprite-Daten abgelegt, denn ungenutzter Speicher wird so noch sinnvoll genutzt. Doch was passiert, wenn nun plötzlich etwas von der Datasette geladen werden soll? Nun, Sie könnten sich damit abfinden, daß Ihre Daten verloren sind. Das jedoch muß nicht sein. Wieso sollen wir dem C64 nicht einfach vortäuschen, daß der Kassettenpuffer bei einer anderen Adresse beginnt? Nichts einfacher als das:

Nehmen wir einmal an, wir wollen den Kassettenpuffer dort beginnen lassen, wo sonst der Bildschirmspeicher beginnt. Das ist recht sinnvoll, weil er beim Laden oder Speichern in der Regel sowieso nicht benutzt wird. Geben Sie bitte folgende Befehle ein:

```
POKE 178,0
POKE 179,4
```

Der Kassettenpuffer beginnt nun bei Adresse 1024. Der Trick ist im Grunde recht einfach. In den Speicherstellen 178/179 befindet sich ein Zeiger, der auf den Beginn des Kassettenpuffers zeigt. Die Adresse 178 enthält normalerweise den Wert 60, und die Adresse 179 enthält normalerweise den Wert 3. Daraus ergibt sich die Anfangsadresse des Kassettenpuffers: 828.

Soll er nun bei 1024 - dem Bildschirmspeicher - beginnen, so müssen wir nur die beiden Werte dieser Adressen ändern. Es ist selbstverständlich auch eine andere Adresse für den Kassettenpuffer möglich. Soll er z.B. bei 49152 beginnen, so müssen wir den Anfangswert nur zerlegen (siehe Anhang) und in die oben genannten Adressen schreiben. Mit

```
POKE 178,0
POKE 179,192
```

begint der Kassettenpuffer bei der Adresse 49152. Sie können also auch ruhig während der Arbeit mit der Datasette irgendwelche Daten im ursprünglischen Kassettenpuffer ablegen!

2.2 Positive Zahl bei FRE

Wer kennt es nicht, das allgegenwärtige Problem: Sie haben gerade ein Programm geschrieben und wollen nun wissen, wieviel Speicherplatz Ihnen noch zur Verfügung steht. Also geben Sie kurz

```
PRINT FRE(0)
```

ein, und Sie sehen:

```
-26681
```

Moment mal, da kann doch etwas nicht stimmen! Hat sich etwa der C64 Speicher geliehen oder warum steht er in den roten Zahlen? Das ganze Problem liegt darin, daß der C64 eine Integerzahl ausgibt. Eine Integerzahl kann - auf Grund der vorherigen Umrechnung ins Fließkommaformat, wobei ein Umrechnungsfehler entsteht - nur Werte zwischen -32768 und

32767 annehmen. Um negative Werte in die 'richtige' Zahl umzurechnen, muß man 65535 addieren. Nach der Ausgabe von -26681 und dem Addieren ergibt sich nun ein freier Speicherplatz von

```
-26681+65535=38854
```

Byte. Sie haben in diesem Fall also noch genug Platz!

2.3 Verändern der FRE-Routine

Die oben gezeigte Lösung zum Fehler in der FRE-Funktion ist zwar recht einfach, doch sobald man öfter den noch vorhandenen Speicherplatz wissen möchte, wird es ganz schön lästig, immer - wenn die ausgegebene Zahl negativ war - die Zahl 65535 zu addieren. Deshalb möchte ich Ihnen hier nun ein kleines Programm vorstellen, das die Routine so abändert, daß immer eine positive Zahl ausgegeben wird.

```
10 FOR A=40960 TO 49151
20 POKE A,PEEK (A)
30 NEXT A
40 FOR A=48981 TO 48997
50 READ B
60 POKE A,B
70 NEXT A
80 FOR A=45965 TO 45968
90 READ B
100 POKE A,B
110 NEXT A
120 POKE 1,54
130 DATA 165,52,229,50,162,0,134,13,133,98,132,99
140 DATA 162,144,76,73,188,76,85,191,234
```

Zuerst wird der Bereich von 40960 bis 49151 ins darunterliegende RAM kopiert. Danach wird ein kleines Maschinenprogramm in dem Bereich von 48981 bis 48997 abgelegt. Sie sollten nun darauf achten, daß ein Laden oder Speichern auf/von Datasette diese Routine zerstört, was einen Absturz des Rechners

beim nächsten Aufruf der FRE-Routine bedeuten würde. Möchten Sie dennoch die Datasette benutzen, so setzen Sie den Kassettenpuffer an eine andere Stelle (siehe Kapitel 2.1).

Danach wird ein Teil des kopierten ROM geändert, so daß nun der fehlerhafte Teil übersprungen wird. Zum Schluß wird dem Rechner noch mitgeteilt, daß er nun die ROM-Routinen im RAM findet.

2.4 Kopieren des ROM ins RAM

Das Ziel jedes interessierten Programmierers ist es wohl, das Betriebssystem seinen eigenen Wünschen anzupassen, es also abzuändern. Doch leider befindet sich das Betriebssystem im ROM, was ja bekanntlich nicht zu ändern ist. Es muß zur Bearbeitung erst ins RAM kopiert werden. Im folgenden werde ich Ihnen zwei Routinen vorstellen, die das Kopieren des ROM in das RAM für uns übernehmen. Folgende Routine kopiert einen beliebigen ROM-Bereich ins RAM. Sie ist in BASIC geschrieben und daher etwas langsam, jedoch gut zu durchschauen.

```
1 FOR A=40960 TO 49151
2 POKE A,PEEK (A)
3 NEXT A
4 POKE 1,54
```

Das Prinzip ist recht einfach: In einer FOR-NEXT-Schleife werden die Werte des ROM ausgelesen und in das darunterliegende RAM geschrieben. Anschließend wird dem Rechner mitgeteilt, daß das Betriebssystem nun im RAM ist.

2.5 Verschieben des Bildschirmspeichers

Der Bildschirmspeicher ist ein Bereich von 1000 Byte, der die ASCII-Werte der Zeichen enthält, die Sie auf dem Bildschirm sehen können. Er beginnt bei der Adresse 1024 und endet bei 2023, denn es müssen 25*40 Zeichen zwischengespeichert werden.

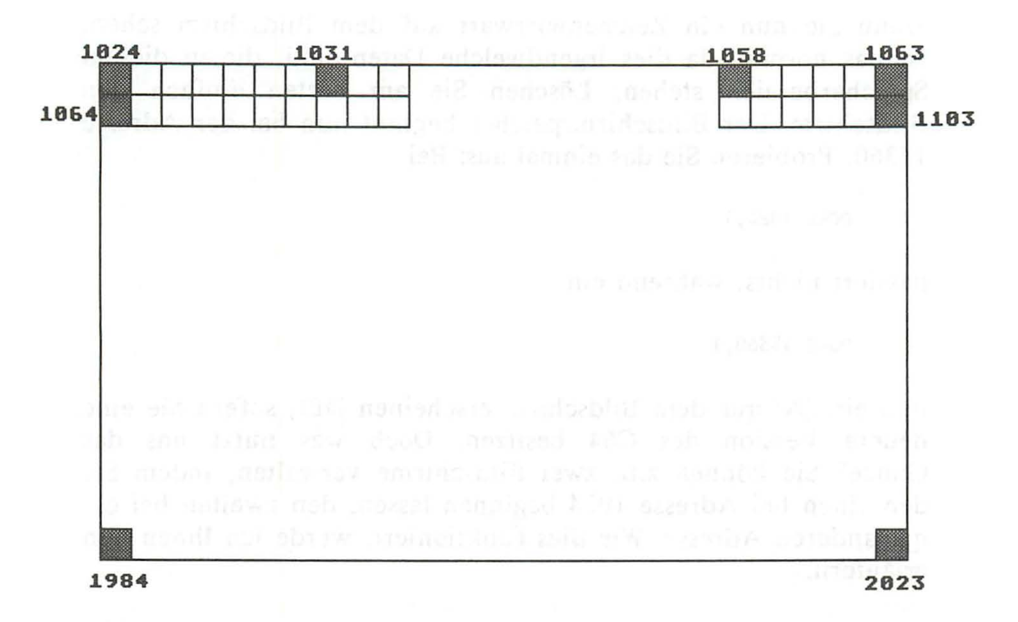


Abb. 1: Bildschirmspeicher

Das läßt sich leicht überprüfen. Geben Sie einmal

```
POKE 1024,1
```

ein. In der linken oberen Ecke des Bildschirms erscheint daraufhin ein A, denn 1 ist der ASCII-Wert für "A". So können Sie den Bildschirm auf andere Weise füllen, was jedoch etwas umständlich ist. Dies funktioniert jedoch nur bei neueren C64. Sollten Sie einen alten C64 besitzen, so sollten Sie sich nicht wundern, wenn nach Eingabe des obigen Befehls kein A erscheint. Wie die Überschrift dieses Kapitels schon sagt, wollen wir nun den Bildschirmspeicher verschieben, ihn also bei einer anderen Adresse beginnen lassen. Geben Sie einmal folgendes ein:

```
POKE 53272,(PEEK (53272) AND 15) OR 240 : POKE 648,15360/256
```

Wenn Sie nun ein Zeichenwirrwarr auf dem Bildschirm sehen, ist das normal, da dies irgendwelche Daten sind, die in diesem Speicherbereich stehen. Löschen Sie am besten einfach den Bildschirm. Der Bildschirmspeicher beginnt nun bei der Adresse 15360. Probieren Sie das einmal aus: Bei

```
POKE 1024,1
```

passiert nichts, während ein

```
POKE 15360,1
```

nun ein "A" auf dem Bildschirm erscheinen läßt, sofern Sie eine neuere Version des C64 besitzen. Doch was nützt uns das Ganze? Sie können z.B. zwei Bildschirme verwalten, indem Sie den einen bei Adresse 1024 beginnen lassen, den zweiten bei einer anderen Adresse. Wie dies funktioniert, werde ich Ihnen nun erläutern.

In der Adresse 53272 (Register 24 des VIC) bestimmen die Bits 4-7, wo der Bildschirmspeicher liegt. Haben Sie den Wert 0, so liegt der Bildschirmspeicher ab 0, bei Wert 1 liegt er ab 1024 (dies ist der Normalwert), bei Wert 2 bei 2048 usw. Sind die letzten 4 Bits gesetzt, haben sie also den Wert 16, so liegt der Bildschirmspeicher ab 15360, was zugleich unter normalen Umständen den höchsten Wert darstellt.

Mit einem Trick lassen sich jedoch auch 'höhere' Speicherbereiche vom VIC ansprechen, worauf ich hier nicht eingehen möchte. Der Bildschirmspeicher kann nur in 1024er Schritten verschoben werden. Die richtigen Werte für die Adresse 1024 erhalten Sie durch folgenden Befehl:

```
POKE 53272,(PEEK (53272) AND 15) OR (Anfang/1024)*16
```

Nun muß nur noch in Speicherstelle 648 angegeben werden, ab welcher Adresse sich der Bildschirmspeicher befindet. Den richtigen Wert für diese Speicherstelle erhalten Sie, indem Sie die Anfangsaddresse durch 256 teilen. Durch

```
POKE 648,Adresse/256
```

wird dem Computer der neue Wert mitgeteilt. Es ist nun ganz einfach, zwei Bildschirme gleichzeitig zu verwalten. Nehmen wir einmal an, Sie müssen in Ihrem Statistikprogramm eine Statistik aufbauen lassen, was jedoch etwas länger dauert. Es wäre nun nicht schön, wenn der Benutzer sehen würde, wie sie langsam aufgebaut wird. Der trickreiche Programmierer - und diese Tricks soll dieses Buch Ihnen ja verraten - baut die Statistik im Hintergrund auf, während der Bildschirm gerade etwas anderes anzeigt.

Wir müssen also erst einmal einen Anfangswert für den zweiten Speicherbereich auswählen. In dem gleich folgenden Beispiel ist dies die Addresse 15360. Nun soll dem Rechner mitgeteilt werden, daß alle Ausgabebefehle schon den zweiten Bildschirm betreffen, wodurch der erste unberührt bleibt. Dies geschieht durch:

```
POKE 648,15360/256.
```

Nun können Sie beliebig 'printen', ohne daß der Benutzer davon etwas merkt, denn angezeigt wird immer noch der erste Bildschirm, während der zweite verändert wird. Ist das zweite Bild fertig aufgebaut, so muß nur noch durch

```
POKE 53272,(PEEK (53272) AND 15) OR (15360/1024)*16
```

angemeldet werden, daß der Bildschirmspeicher nun bei 15360 beginnt. Dies hört sich alles recht kompliziert an. Aus diesem Grunde habe ich ein kleines Demoprogramm geschrieben, um das einmal in der Praxis zu zeigen:

```
10 PRINT CHR$(147);"Sie sehen nun den ersten Bildschirm,"
20 PRINT "während der 2. aufgebaut wird"
30 POKE 648,15360/256
40 PRINT CHR$(147)
50 FOR A=1 TO 24
60 PRINT "2. Bildschirm"
70 FOR B=1 TO 100:NEXT B
80 NEXT A
90 POKE 53272,(PEEK (53272) AND 15) OR (15360/1024)*16
```

3. Die Floppy 1541

Die meisten C64-Besitzer werden wohl auch eine Floppy besitzen, da dieses Speichermedium um einiges bequemer und leichter zu bedienen ist als die Datasette. Doch nicht nur diesbezüglich ist sie der Datasette überlegen; es lassen sich nämlich auch viel mehr Dinge mit ihr 'anstellen', deshalb einige Tips und Tricks dazu auf den folgenden Seiten.

3.1 Laden und Speichern

Der erste Teil dieses dritten Kapitels befaßt sich mit dem Speichern und dem Laden, denn dieses Thema kann sehr variantenreich sein.

3.1.1 Abfrage eines Kennwortes

Wenn Sie nach der Lektüre des Kapitels 1.2 wieder Ihre Datasette rauskramen, um auch Ihre Programme vor unbefugten Benutzern zu schützen, so kann ich Ihnen die erfreuliche Mitteilung machen, daß es auch bei der Floppy möglich ist, Programme nur dann starten zu lassen, wenn vorher beim Programmnamen eine bestimmte Kennung angegeben wurde.

Im Gegensatz zur Datasette gibt es jedoch keinen 'Floppypuffer', in dem Daten zwischengespeichert werden. Es gibt jedoch einen Zeiger bei Addresse 187/188, der auf den zuletzt genutzten Programmnamen bei Floppy-Operationen zeigt. Sie können sich dies zunutze machen und diesen Namen durch folgendes Programm abfragen:

```
10 A = PEEK (187) + PEEK (188)*256
20 FOR B = A TO 40959
30 C = PEEK (B)
40 NAME$ = NAME$ + CHR$(C)
50 NEXT B
60 PRINT NAME$
```

Das Prinzip ist recht einfach. Der Zeiger zeigt auf den Namen, der dann in einer Schleife Zeichen für Zeichen eingelesen und ausgegeben wird. Das Prinzip einer Abfrage nach einer Kennung ist einfach: Der Benutzer muß beim Laden an das Programm noch eine bestimmte Kennung anhängen, die dann durch das Programm abgefragt wird. Die einzige Bedingung ist, daß das Programm unter einem 16 Buchstaben langen Namen abgespeichert wird. Hier nun die Abfrage:

```
10 ADR = PEEK (187) + PEEK (188)*256
20 FOR A = ADR+16 TO ADR+18
30 READ A$
40 ZEICHEN = ASC (A$)
50 IF ZEICHEN = PEEK (A) THEN 70
60 SYS64738
70 NEXT A
80 DATA O,K,!
```

Speichern Sie dieses Programm unbedingt unter einem 16 Zeichen langen Namen ab, und laden Sie es dann ohne Kennwort. Es kommt zu einen Reset. Laden Sie nun das Programm noch einmal, und hängen Sie die Endung OK! an den Namen. Daraufhin arbeitet das Programm nach dem Laden normal. Kombiniert man diesen Trick mit einen Listschutz, so ist es für Unbefugte nicht einfach, dieses Programm zu starten.

3.1.2 Ermitteln der Gerätenummer

Jedes Peripheriegerät besitzt eine Kennungsnummer, die sogenannte Gerätenummer. Diese Nummer ist dazu da, um z.B. den Drucker von der Floppy zu unterscheiden. Schreiben Sie ein Programm, das mit verschiedene Geräten (z.B. Floppy oder Datasette) arbeiten soll, so sollte das Programm ermitteln können, welche Geräte der Benutzer angeschlossen hat. Dies wäre natürlich auch durch eine Abfrage im Programm möglich, doch ist dies weniger elegant. Die Adresse 186 enthält die Gerätenummer des zuletzt benutzten Gerätes. Durch

```
PRINT PEEK (186)
```

können Sie diese erfahren. So könnten zum Beispiel die Befehle zum Nachladen eines Programmteiles folgendermaßen lauten:

```
10 OPEN 1,8,15
20 NUMMER = PEEK (186)
30 LOAD "PRG.",NUMMER
```

Ich habe hier die Zeile 10 eingefügt, damit gewährleistet ist, daß sich der nachfolgende Befehl auch auf die Floppy bezieht. Wenn Sie das Programm speichern, ist das nicht mehr nötig, denn dann ist es gewährleistet, daß in der Speicherstelle 186 der Wert 8 steht. Würde Zeile 10 wegfallen und hätten Sie vorher noch nicht mit der Floppy gearbeitet, so ergäbe dies einen Fehler. Arbeiten Sie mit der Datasette, so müssen Sie die Zeile 10 entsprechend ändern. Auch das Öffnen eines Floppykanals - um z.B. eine sequentielle Datei zu öffnen - ist so möglich. Das Prinzip ist das gleiche:

```
10 OPEN 1,8,15
20 NUMMER = PEEK (186)
30 OPEN 1,NUMMER,15
```

So ist es möglich, daß ein Dateiverwaltungsprogramm sowohl mit der Floppy als auch mit der Datasette zusammenarbeitet. Wie oben schon erwähnt, gibt diese Speicherstelle nur über das zuletzt benutzte Gerät Auskunft, jedoch nicht über alle angeschlossenen Geräte. Es ist also hier Vorsicht geboten, denn sonst kann es passieren, daß Sie ein Programm von Diskette nachladen wollen, vorher aber den Drucker benutzt haben, wodurch nun das Programm versuchen würde, etwas vom Drucker zu laden, was natürlich zu einem Fehler führen würde! Die sicherste Methode ist es, direkt am Programmanfang diese Nummer abzufragen, was den Vorteil hat, daß diese dann nicht mehr verändert wird.

3.1.3 Direktes Abspeichern von Maschinenprogrammen

Haben Sie für Ihre Maschinenroutinen auch immer einen BASIC-Loader geschrieben, der das Maschinenprogramm dann durch einen SYS-Befehl aufgerufen hat? Diese Methode ist nicht

nur zeitaufwendiger, sondern verbraucht auch mehr Speicherplatz. Außerdem wird auf diese Weise immer das BASIC-Programm zerstört, das im Speicher ist. Eine einfachere Methode ist es, den Speicherbereich direkt abzuspeichern. Dazu kurz einige Grundlagen zum SAVE-Befehl:

Der SAVE-Befehl liest die Werte für den Programmanfang und das Programmende aus den Speicherstellen 43-46. Danach speichert er den so erhaltenen BASIC-Bereich ab. Wollen Sie nun einen beliebigen Speicherbereich abspeichern, so müssen Sie lediglich die Werte der Adressen 43-46 so ändern, daß der Zeiger in den Adressen 43/44 auf den Anfang des abzuspeichernden Bereiches zeigt und der Zeiger in den Adressen 45/46 auf das Ende des abzuspeichernden Bereiches. Nehmen wir einmal an, Sie wollen den Speicherbereich von 4000-5000 abspeichern. Dies geschieht durch die Befehle:

```
POKE 43,160
POKE 44,15
POKE 45,136
POKE 46,19
CLR
SAVE "PRG.NAME",8,1
```

Die Eins hinter dem SAVE-Befehl bedeutet, daß das Programm absolut abgespeichert wird, das heißt, daß es beim Laden direkt an die richtige Adresse geladen wird. Ersetzen Sie beim SAVE-Befehl die Acht durch eine Eins, so funktioniert dieser Trick auch mit der Datasette. Möchten Sie nach dem Abspeichern noch weiter in BASIC programmieren, so müssen Sie die Zeiger wieder richtig setzen. Dies geschieht durch folgende Befehle:

```
POKE 43,1
POKE 44,8
POKE 45,3
POKE 46,8
```

Durch diesen Trick lassen sich nicht nur Maschinenprogramme abspeichern; es ist auch möglich, in Verbindung mit dem Trick aus Kapitel 2.4 verschiedene Bildschirme zu verwalten und diese

nachzuladen. So läßt sich ein Bildschirm, der z.B. ab 15360 liegt, durch folgende Befehle abspeichern:

```
POKE 43,0
POKE 44,60
POKE 45,0
POKE 46,64
CLR
SAVE "BILDSCHIRM_2",8,1
```

Hier sei wieder angemerkt, daß dies nur bei neueren Betriebssystemversionen funktioniert. Auch der Zeichensatz läßt sich so abspeichern, wodurch Ihre Programme um einiges flexibler werden, denn welche Programme arbeiten schon mit verschiedenen Zeichensätzen? Hier nun noch ein zusätzlicher Trick zur Datasette. Angenommen, Sie haben einen Bildschirm ab 15360 abgespeichert, Sie wollen ihn jedoch nun an einer anderen Stelle haben. Auch dies ist kein Problem für Datasetten-Besitzer. Geben Sie einfach den Befehl

```
SYS 63276
```

ein, und es erscheint die Meldung

```
PRESS PLAY ON TAPE
```

die Sie nun auch befolgen sollten. Daraufhin wird jedoch nicht das ganze Programm, sondern nur der Tape-Header geladen. Im Tape-Header stehen die Werte, von wo bis wo das Programm im Speicher stehen soll. Sie können diese Werte durch folgende POKEs abändern:

```
POKE 829,ANFANG WERT1
POKE 830,ANFANG WERT2
POKE 831,ENDE WERT1 + 2
POKE 832,ENDE WERT2
```

Setzen wir einmal den Bildschirm an Adresse 8192-9191. Dazu müssen Sie folgende POKEs angeben:

```
POKE 829,0
POKE 830,32
POKE 831,0
POKE 832,34
```

Nun müssen Sie nur noch den Ladevorgang durch

```
SYS 62849
```

fortsetzen, und anschließend den Befehl

```
NEW
```

eingeben. Nun beginnt Ihr Bildschirm ab der Speicherstelle 8192.

3.1.4 Vertauschen von SAVE und LOAD

Ein kleiner Kopierschutz läßt sich schon durch ein paar POKEs verwirklichen! Für viele Befehle oder Routinen gibt es Zeiger, die auf die Anfangsadressen der Routinen zeigen. Auch für die Befehle SAVE und LOAD gibt es solche Zeiger. Der Zeiger für SAVE befindet sich bei Adresse 818/819, der LOAD-Zeiger befindet sich bei 816/817. Diese beiden Zeiger lassen sich nun vertauschen, so daß bei Eingabe von SAVE "Prg.name",8 nicht wie gewöhnlich gespeichert, sondern der Befehl VERIFY wird ausgeführt. Auch läßt sich der LOAD-Zeiger so verändern, daß nicht geladen sondern gespeichert wird. Dieses Austauschen der Zeiger übernimmt folgendes Programm:

```
10 LL = PEEK (816)
20 LH = PEEK (817)
30 POKE 816,PEEK (818)
40 POKE 817,PEEK (819)
50 POKE 818,LL
60 POKE 819,LH
```

Die Funktionsweise des Programmes ist recht simpel. In Zeile 10 und 20 wird der Zeiger des SAVE-Befehls zwischengespeichert. Dann erhält der SAVE-Zeiger die Werte des LOAD-Zeigers und der LOAD-Zeiger die vorher zwischengespeicherten Werte des SAVE-Zeigers.

Sie können diese POKEs auch im Direktmodus eingeben. Die 'normalen' Werte des SAVE-Zeigers sind 165/244 ($F5ED) und die des LOAD-Zeigers sind 237/245 ($F4A5). Richtig zur Verzweiflung bringen kann man jemanden, wenn man den LOAD-Zeiger so läßt, wie er ist, und den SAVE-Zeiger einfach auf den LOAD-Befehl zeigen läßt. Nun läßt sich überhaupt nichts mehr abspeichern.

Es lassen sich jedoch nicht nur diese beiden Zeiger untereinander vertauschen; man kann sie auch auf ganz andere Routinen oder Befehle zeigen lassen. Folgende POKEs bewirken, daß der SAVE-Zeiger auf die Adresse 64738 zeigt, wodurch bei jedem Aufruf von SAVE ein RESET ausgeführt wird:

```
POKE 818,226
POKE 819,252
```

Wer es nicht ganz so radikal mag (wer anderen eine Grube gräbt, fällt selbst hinein!), kann den SAVE-Zeiger ja einfach auf ein RTS zeigen lassen. RTS ist ein Maschinensprachebefehl und bewirkt nichts anderes als einen Rücksprung zu BASIC, was sich durch eine READY-Meldung äußert. Hier nun die beiden Befehle:

```
POKE 818,26
POKE 819,167
```

Alles hier Genannte läßt sich natürlich auch auf den LOAD-Zeiger anwenden!

3.1.5 Abspeichern eines Programmes als SEQ-File

Speichern Sie ein Programm mittels

```
SAVE "PRG.NAME",8
```

ab, so bekommt dieses File im Directory die Endung PRG. Wollen Sie jedoch, daß Ihr Programm eine andere Endung be-

kommt, so ist es auch möglich, es als sequentielles File oder als User-File abzuspeichern. Dazu müssen Sie beim Abspeichern folgendes eingeben:

```
SAVE "PRG.NAME,ENDUNG,STATUS",8
```

Hier muß nun die Variable ENDUNG durch die Abkürzung für den Filetyp ersetzt werden. Hier nun eine Auflistung der möglichen Filetypen und deren Abkürzungen:

SEQ	Sequentielle Datei
USR	User-Datei

Die Variable STATUS gibt an, ob das Programm gelesen oder geschrieben werden soll. Soll auf Diskette geschrieben werden, so muß an dessen Stelle ein W für WRITE stehen, soll gelesen werden, so muß an dessen Stelle ein R für READ stehen. Wollen Sie also ein Programm namens Programm1 als sequentielles File abspeichern, so müssen Sie folgendes eingeben:

```
SAVE "PROGRAMM1,S,W",8
```

Noch einmal zur Verdeutlichung: S gibt an, daß das Programm als sequentielles File abgespeichert wird, das W gibt an, daß das File geladen wird. Wollen Sie es nun wieder laden, so geben Sie bitte folgendes ein:

```
LOAD "PROGRAMM1",8
```

Sie werden Pech haben. Der Computer gibt einen FILE NOT FOUND ERROR aus, denn er sucht ja nach einem PRG-File, das den Namen Programm1 hat. Dies existiert jedoch nicht, denn wir haben das Programm ja als sequentielles File abgespeichert. Es muß also auch wieder als sequentielles File geladen werden. Geben Sie deshalb bitte

```
LOAD "PROGRAMM1,S,R",8
```

ein, und das Programm wird geladen. Auch hier noch einmal die kurze Erklärung: Das S gibt an, daß ein sequentielles File gela-

den werden soll, und das R gibt an, daß dieses File gelesen werden soll. Wollen Sie ein Programm als User-File abspeichern, so geben Sie bitte

```
SAVE "PROGRAMM2,U,W",8
```

ein. Im Directory erscheint daraufhin

```
"PROGRAMM2"        USR
```

Wollen Sie es wieder laden, so müssen Sie

```
LOAD "PROGRAMM2,U,R",8
```

eingeben. Nun, was hat das Ganze für einen Sinn? Wollen Sie ein Programm ohne großen Umstand vor der Benutzung Unbefugter schützen, so ist dies eine recht sichere Methode, vorausgesetzt, der Trick ist dem unbefugten Benutzer unbekannt. Denn wer vermutet schon hinter einem sequentiellen File ein BASIC-Programm? Und selbst wenn er versucht, es normal zu laden, wird er durch einen FILE NOT FOUND ERROR darauf aufmerksam gemacht, daß das Programm im Grunde nicht existiert.

3.1.6 Automatisches Laden und Starten von Programmen

Haben Sie bis jetzt auch immer Ihre Programme mit

```
LOAD "PRG.NAME",8
```

geladen und anschließen mittels

```
RUN
```

gestartet? Daß dies jedoch auch einfacher geht, zeigt Ihnen dieser Trick: Bekanntlich kann ein Programm von der Datasette durch Drücken der <SHIFT/RUN-STOP>-Tasten geladen und automatisch gestartet werden. Solch eine Tastenkombination gibt es leider nicht für die Floppy. Geben Sie jedoch einmal folgende Befehlsreihe ein:

```
LOAD "PRG.NAME",8:
```

Drücken Sie jedoch noch nicht <RETURN>! Wenn sich der Cursor hinter dem Doppelpunkt befindet, so drücken Sie die Tastenkombination <SHIFT/RUN-STOP>. Nun wird das Programm geladen, und direkt danach wird es durch RUN automatisch gestartet. Wie funktioniert dieser Trick? Nun, der LOAD-Befehl ignoriert eventuelle Befehle, die nach dem Doppelpunkt stehen. Das Drücken der Tasten <SHIFT/RUN-STOP> bewirkt nichts anderes als die Eingabe von

```
LOAD
RUN
```

Da der LOAD-Befehl nun den Befehl, der nach dem Doppelpunkt kommt, ignoriert, wird der zweite LOAD-Befehl einfach ignoriert. Danach folgt jedoch noch der Befehl

```
RUN
```

wodurch das Programm letztendlich gestartet wird. Sie sehen: Dies ist wieder einer der recht kurzen und simplen Tricks, auf die man jedoch erst einmal kommen muß. Gewöhnen Sie es sich jedoch an, Ihre Programme immer auf diese Art zu laden, so können Sie sich eine Menge an Arbeit ersparen.

3.2 Tips und Tricks zum Directory

Das Directory - immer da, doch kaum bemerkt. Was sich alles damit anfangen läßt, zeigt Ihnen dieses Kapitel.

3.2.1 Verstecktes Directory

Manche professionelle Programme haben ein geschütztes Directory, das sich nicht auflisten läßt. Dies hat den Vorteil, daß sich die einzelnen Files nicht laden lassen, da man deren Namen nicht weiß. Dadurch ist es schwerer, Programme dieser Diskette zu kopieren. Ich möchte Ihnen in diesem Kapitel eine Routine vorstellen, die das Directory vor unbefugten 'Einsehern' schützt.

Um eine Diskette so zu schützen, geben Sie bitte dieses Programm ein und starten sie es. Am besten nehmen Sie erst einmal eine Diskette, die keine wichtigen Daten enthält, damit Sie die Funktion dieses Programm kennenlernen. Hier nun das Listing:

```
10 OPEN 1,8,3,"#"
20 PRINT# 1,CHR$(20);CHR$(20);CHR$(20);
30 PRINT# 1,CHR$(0);CHR$(0);CHR$(0)
40 OPEN 2,8,15,"B-P3,144"
50 PRINT# 2,"U2:3,0,18"
60 PRINT# 2,"I"
70 CLOSE 1
80 CLOSE 2
```

Während des Auflistens des Directorys wird jedesmal überprüft, ob drei Nullbytes eingelesen wurden. Ist dies der Fall, so wird das Auflisten unterbrochen. Genau nach diesem Prinzip arbeitet dieses Programm auch. Es schreibt an den Anfang des Directorys drei Nullen, was dann beim Listen sofort zum Abbruch führt, denn der Rechner liest zuerst drei Nullbytes ein - die Kennung für das Ende des Directorys.

Dieses Programm ist jedoch nicht ganz ungefährlich, und es sollte nur mit Obacht angewendet werden, denn es wird nicht nur das Directory vor dem Auflisten geschützt, sondern die Diskette wird auch vor dem Beschreiben geschützt. Sie sollten dieses Programm also nie auf Datendisketten oder sonstigen Disketten, die oft beschrieben werden, anwenden. Die Aufhebung dieses Programmes ist nur durch ein Neuformatieren möglich, womit die Daten jedoch restlos verlorengehen.

3.2.2 Schützen eines Teiles des Directorys

Möchten Sie nur einen Teil des Directorys löschen, so ist das Programm aus Kapitel 3.2.1 ungeeignet, da immer das ganze Directory geschützt wird. Außerdem kann so auf die Diskette nichts mehr geschrieben werden, was von Vorteil, aber auch ein Nachteil sein kann. Es kann aber nach dem gleichen Prinzip auch nur ein Teil des Directorys geschützt werden, indem Sie

einfach einen Programmnamen so umbenennen, daß er mit drei Nullbytes endet. Nehmen wir einmal an, Sie haben folgendes Directory:

```
0  "TIPS&TRICKS"       AP
10 "PROGRAMM1"         PRG
10 "PROGRAMM2"         PRG
10 "PROGRAMM3"         PRG
10 "PROGRAMM4"         PRG
10 "PROGRAMM5"         PRG

596 BLOCKS FREE.
```

Wenn Sie nun Ihr Directory ab PROGRAMM3 schützen wollen, so müssen Sie an den Namen PROGRAMM2 drei Nullbytes anhängen. Dies können wir mit dem RENAME-Befehl erreichen:

```
OPEN 1,8,15,"R:PROGRAMM2"+CHR$(0)+CHR$(0)+CHR$(0)+"=PROGRAMM2"
CLOSE 1
```

Wird nun das Directory eingelesen und anschließend aufgelistet, so wird nach der Anzeige von PROGRAMM2 aufgehört, denn der Rechner erkennt, daß hier drei Nullbytes folgen, was für ihn das Ende des Directorys bedeutet. Nach Eingabe von

```
LOAD "$",8
LIST
```

erscheint folgende Ausgabe:

```
0  "TIPS&TRICKS"       AP
10 "PROGRAMM1"         PRG
10 "PROGRAMM2"         PRG

596 BYTES FREE
```

Dies hat einige Vorteile gegenüber dem Trick aus Kapitel 3.2.1: Erstens ist die Diskette nicht schreibgeschützt, was mitunter ganz nützlich ist, und dieser Trick ist wieder umzukehren, d.h. Sie können Ihr Directory wieder 'entschützen'.Directory schützen. Um wieder das ganze Directory sehen zu können, müssen die drei Nullbytes aus dem abgeänderten Programmnamen 'verschwinden'. Dies geschieht durch den RENAME-Befehl:

```
OPEN 1,8,15,"R:PROGRAMM2=PROGRAMM2"+CHR$(0)+CHR$(0)+CHR$(0)
CLOSE 1
```

Dadurch werden die Nullbytes aus dem PROGRAMM2 entfernt, wodurch das Directory weiter aufgelistet wird. Nach Eingabe von

```
LOAD"$",8
LIST
```

erscheint wieder das ganze Directory:

```
0  "TIPS&TRICKS"      AP
10 "PROGRAMM1"        PRG
10 "PROGRAMM2"        PRG
10 "PROGRAMM3"        PRG
10 "PROGRAMM4"        PRG
10 "PROGRAMM5"        PRG
596 BLOCKS FREE.
```

3.2.3 Steuercodes im Directory

Haben Sie schon einmal versucht, im Directory Steuercodes unterzubringen? So komisch das auch klingen mag, es ist möglich! Speichern Sie bitte einmal ein Programm folgendermaßen ab:

```
SAVE CHR$(147)+CHR$(18)+CHR$(158)+"PRG.NAME",8
```

Wenn Sie es nun laden, wird nach der Meldung

```
SEARCHING
```

erst der Bildschirm gelöscht, dann der REVERS-Mode angeschaltet und anschließend die Meldung

```
LOADING PRG.NAME
```

in inverser Schrift ausgegeben. Wie geht das? Nun, hat die Floppy das Programm 'gefunden', so gibt sie eine Meldung und den Programmnamen aus. Da der Programmname aber mit

Steuercodes beginnt, werden diese erst ausgeführt, bevor der eigentliche Name erscheint. Es ist auch beim Abspeichern möglich, die Steuercodes direkt durch Drücken bestimmter Tasten zu erzeugen, was etwas Tipparbeit spart. Geben Sie z.B. folgenden Befehl ein:

```
SAVE "<SHIFT/HOME><CRTL/1>PRG.NAME",8
```

Die in den Klammern stehenden Ausdrücke stellen hierbei die zu drückenden Tastenkombinationen dar. Beim nächsten Laden wird darauf erst der Bildschirm gelöscht und dann die Schriftfarbe auf schwarz gewechselt. Im folgenden finden Sie eine Tabelle der wichtigsten Steuercodes aufgeführt:

CHR$(5)	Weiß
CHR$(8)	Commodore-Taste blockiert
CHR$(9)	Commodore-Taste entriegelt
CHR$(14)	Umschaltung auf Kleinschrift
CHR$(18)	Reverse Mode
CHR$(19)	Home
CHR$(20)	Delete
CHR$(28)	Rot
CHR$(30)	Grün
CHR$(31)	Blau
CHR$(142)	Umschaltung auf Großschrift
CHR$(144)	Schwarz
CHR$(147)	Bildschirm löschen
CHR$(156)	Purpur
CHR$(158)	Gelb
CHR$(159)	Cyan

Durch den Character-Code 20 ist es möglich, das Anzeigen des Programmnamens zu unterbinden. Es müssen nur beim Abspeichern hinter dem Programmnamen so viele CHR$(20) angehängt werden, wie der Programmname lang ist.

3.2.4 Sondereinträge im Directory

Normalerweise zeigt das Directory an, welche Files sich auf der Diskette befinden. Es werden deren Namen in Anführungsstrichen angezeigt. Es besteht jedoch auch die Möglichkeit, die Namen durch Bemerkungen zu erweitern. So ist es z.B. möglich, daß sich ein Programm unter folgendem Namen im Directory befindet:

```
10 "PROGRAMM1" V1.0    PRG
```

Das Besondere hierbei ist jedoch, daß die Erweiterung nicht beim Laden eingegeben werden muß. Das Programm läßt sich ganz normal durch

```
LOAD "PROGRAMM1",8
```

laden. Solche Erweiterungen im Programmnamen können mitunter recht nützlich sein, so kann die Endung DAT z.B. angeben, daß das Programm nur mit der Datasette läuft, oder die Endung PRT zeigt an, daß man für die Benutzung des Programmes einen Drucker braucht. Weiß der Benutzer das schon vor dem Programmstart, so kann er eventuellen Programmabstürzen vorbeugen.

Auch können alle Programme mit einer Endung versehen werden, die die Art des Programmes kennzeichnet. Anwenderprogramme können so z.B. die Endung ANW haben, während Spiele die Endung SPIEL haben. (Voraussetzung dafür ist jedoch, daß der Programmname und die Endung nicht länger als 16 Buchstaben sind). Auch die Kennzeichnung der aktuellen Programmversion ist so gut möglich, wenn man ein Programm über eine längere Zeit programmiert. Der Trick ist recht einfach. Geben Sie einmal folgende Befehle ein:

```
OPEN 1,8,15,"R:PROGRAMM1"+CHR$(160)+" T&T"+"=PROGRAMM1"
CLOSE 1
```

Vorausgesetzt, es befand sich schon ein Programm mit dem Namen PROGRAMM1 auf der Diskette, so wurde es nun umbenannt und nach der Eingabe von

```
LOAD "$",8
```

erscheint daraufhin:

```
10 "PROGRAMM1" T&T     PRG
```

Geladen wird das Programm jedoch ganz normal mittels:

```
LOAD "PRGRAMM1",8
```

Wie funktioniert dies jedoch? Wie Sie am obigen Beispiel erkennen können, wurde der Programmname einfach umbenannt, und zwar wurde zwischen dem eigentlichen Namen und der Endung der Character-Code 160 eingefügt. CHR$(160) steht für das geschützte Leerzeichen SHIFT/SPACE. Dies zeigt dem Computer an, daß der Filename hier zu Ende ist. Deshalb gibt der Computer hier auch die zweiten Anführungsstriche an.

Der Name ist jedoch in Wirklichkeit noch nicht zu Ende, denn er wurde ja so umbenannt, daß hinter dem Anführungszeichen noch etwas kommt, nämlich die Endung, die zwar auch abgespeichert, aber nicht mehr als Teil des Programmnamens angesehen wird. Es muß jedoch eine Einschränkung gemacht werden. Das Komma darf leider nicht in der Endung erscheinen, da die Floppy dies nicht akzeptiert. Wollen Sie diese Änderung rückgängig machen, so müssen das Programm erneut umbenennen. Um also unserem PROGRAMM1 wieder einen 'normalen' Namen zu geben, müssen Sie

```
OPEN 1,8,15,"R:PROGRAMM1=PROGRAMM1"+CHR$(160)+" T&T"
CLOSE 1
```

eingeben. Wollen Sie eine Endung umbenennen, weil Sie z.B. die Programmversion 1.1 verbessert haben und sie nun die Versionsnummer 1.3 erhalten soll, so sollten Sie folgende Befehle eingeben:

```
OPEN 1,8,15,"R:PROGRAMM"+CHR$(160)+" V1.1"+"=PROGRAMM"+CHR$(160)+"
V1.3"
CLOSE 1
```

Durch diesen Trick ist es recht einfach, die Directorys so zu kennzeichnen, daß man sofort weiß, um welche Programme es sich handelt.

3.2.5 Directory ohne Programmverlust

Wer kennt das nicht: Man hat gerade programmiert, hat das Programm noch nicht gespeichert. Doch ausgerechnet jetzt muß man wissen, ob ein bestimmtes File auf der Diskette ist, und beim Laden des Directorys wird das BASIC-Programm im Speicher leider gelöscht. Man hätte nun natürlich die Möglichkeit, das Programm erst abzuspeichern, dann das Directory der Diskette zu laden und anschließend wieder das Programm zu laden. Doch diese Methode ist recht umständlich. Einfacher geht dies, indem Sie

```
POKE 43,PEEK (45)
POKE 44,PEEK (46)+1
```

eingeben. Nun können Sie das Directory laden, anzeigen lassen und auch ausdrucken. Durch die Eingabe von

```
POKE 43,1
POKE 44,8
```

steht Ihnen das BASIC-Programm erneut zur Verfügung. Dieser Trick ist ganz einfach. Da das Directory immer ab der Adresse eingelesen wird, auf die der BASIC-Anfang-Zeiger zeigt, muß dieser nur so abgeändert werden, daß das Directory nicht das Programm im Speicher überschreibt. Dies geschieht durch die beiden Befehle

```
POKE 43,PEEK (45)
POKE 44,PEEK (46)+1
```

die den BASIC-Anfang hinter das Programmende setzen. Wollen Sie nun wieder mit Ihrem Programm arbeiten, so müssen Sie den BASIC-Anfang 'runtersetzen', und zwar auf die ursprünglichen Werte. Da es nichts gibt, was nur Vorteile hat, müssen wir auch durch diesen Trick einen Nachteil hinnehmen: Da das Programm

an das Ende des BASIC-Programmes gesetzt wird, werden die Variablen gelöscht, die sich normalerweise hinter dem BASIC-Programm befinden. Dies können Sie jedoch umgehen, indem Sie den BASIC-Anfang hinter die Variablen legen. Dies geschieht durch

```
POKE 43,PEEK (49)
POKE 44,PEEK (50)
```

Auch dies kann durch die Befehle

```
POKE 43,1
POKE 44,8
```

wieder rückgängig gemacht werden.

3.2.6 Steuerzeichen im Directory

Genau wie bei Programmnamen, so ist es auch im Directory möglich, Steuerzeichen einzubauen. Dazu brauchen Sie lediglich eine leere Diskette zu formatieren und vor dem Diskettennamen die Steuerzeichen einzugeben. Legen Sie bitte eine unformatierte Diskette ins Laufwerk, und geben Sie folgendes ein:

```
OPEN1,8,15,"N:"+CHR$(13)+CHR$(147)+"TIPS&TRICKS,AP"
CLOSE 1
```

Nach Eingabe von

```
LOAD"$",8
LIST
```

wird der Bildschirm gelöscht, bevor das Disketteninhaltsverzeichnis geladen wird. Dieser Trick funktioniert ähnlich wie der aus Kapitel 3.1.5. Vor dem eigentlichen Namen werden beim Formatieren die Steuerzeichen angegeben. Das CHR$(147) wird Ihnen nun vielleicht schon bekannt vorkommen: Es löscht den Bildschirm. Warum muß jedoch vorher ein CHR$(13) eingegeben werden? CHR$(13) ist der Character-Code für das Drücken der RETURN-Taste.

Es muß erst ein RETURN ausgeführt werden, weil - bevor der Diskettenname ausgegeben wird - eine Null und die Anführungsstriche ausgegeben werden. Würde nun direkt das CHR$(147) folgen, so befände sich der Computer noch im Hochkomma-Modus, in dem Steuerzeichen nicht ausgegeben werden, sondern deren grafische Symbole. Dies können Sie ausprobieren, indem Sie

```
PRINT "
```

eingeben. Der Computer befindet sich nun im Hochkomma-Modus. Drücken Sie nun <CLR/HOME>, so wird ein inverses Herz ausgegeben - der Bildschirm aber wird nicht gelöscht. Um dieses Problem zu umgehen, wird vor dem Steuerzeichen noch das Drücken der RETURN-Taste 'simuliert', wobei in den normalen Darstellungsmodus umgeschaltet wird. Das Steuerzeichen wird nun direkt ausgeführt, was ein Löschen des Bildschirms bedeutet. Danach wird der Name der Diskette und das Inhaltsverzeichnis direkt ausgegeben. Sie können selbstverständlich auch andere Steuerzeichen anstelle des CHR$(147) eingeben. Es gelten grundsätzlich die gleichen wie die aus Kapitel 3.15. Um Ihnen das lästige Umblättern zu ersparen, habe ich hier noch einmal die Liste aufgeführt:

CHR$(5)	Weiß
CHR$(8)	Commodore-Taste blockiert
CHR$(9)	Commodore-Taste entriegelt
CHR$(14)	Umschaltung auf Kleinschrift
CHR$(18)	Reverse-Mode on
CHR$(19)	Home
CHR$(20)	Delete
CHR$(28)	Rot
CHR$(30)	Grün
CHR$(31)	Blau
CHR$(142)	Umschaltung auf Großschrift
CHR$(144)	Schwarz
CHR$(147)	Bildschirm löschen
CHR$(156)	Purpur
CHR$(158)	Gelb
CHR$(159)	Cyan

Hier nun noch einige Beispiele für persönliche Inhaltsverzeichnisse (Bedenken Sie bitte, daß beim Formatieren eventuelle Daten, die sich auf der Diskette befinden, verlorengehen!)

```
OPEN 1,8,15,"N:"+CHR$(13)+CHR$(5)+CHR$(14)+"TIPS&TRICKS,AP"
CLOSE 1
```

Hier wird erst die Schriftfarbe auf weiß gewechselt und dann auf Kleinschrift umgeschaltet, bevor das Directory ausgegeben wird.

```
OPEN 1,8,15,"N:"+CHR$(20)+CHR$(20)+"TIPS&TRICKS,AP"
CLOSE 1
```

Die Null und die ersten Anführungsstriche werden gelöscht, bevor das Directory ausgegeben wird.

```
OPEN 1,8,15,"N:"+CHR$(13)+CHR$(156)+CHR$(8)+"TIPS&TRICKS,AP"
CLOSE 1
```

Die Schriftfarbe wechselt auf Purpur, und das Umschalten zwischen Klein- und Großschrift wird blockiert, bevor das Directory ausgegeben wird. Ich habe Ihnen hier nun einige Beispiele gezeigt, wie Sie Ihr Directory umgestalten können. Probieren Sie dies ruhig ein wenig aus, denn Probieren geht über Studieren!

3.3 Erweitertes DOS

In diesem Teil des dritten Kapitels möchte ich Ihnen einige nützliche Routinen und Tricks vorstellen, die den Befehlssatz des DOS sozusagen erweitern (sofern das von BASIC aus geht).

3.3.1 Auslesen des Fehlerkanals

Wer kennt die Situation nicht: Man möchte eine Floppy-Operation durchführen, doch das Knacken bleibt aus, und statt dessen blinkt das Floppy-Lämpchen. Nach Prüfung auf Schreib- oder

andere Flüchtigkeitsfehler weiß man nicht mehr weiter. Doch warum sollte man nicht einfach die Floppy selbst befragen, warum sie nicht gehorcht. Dazu muß der Fehlerkanal der Floppy geöffnet werden, und bestimmte Parameter müssen eingelesen werden. Folgendes Programm erledigt das für Sie:

```
10 OPEN 1,8,15
20 INPUT#1,NR,FEHLER$,SPUR,SEKT
30 PRINT NR","FEHLER","SPUR","SEKT
40 CLOSE 1
```

Zeile 10 öffnet - wie schon erwähnt - den Fehlerkanal der Floppy. In Zeile 20 werden nun die verschiedenen Werte eingelesen. Dabei geben folgende Variablen folgende Werte an:

NR	Fehlernumer
FEHLER$	Fehlermeldung
SPUR	SPUR
SEKT	SEKTOR

Vielleicht werden Sie sich fragen, warum ich für den Sektor die Variable SEKT, jedoch nicht SEKTOR benutzt habe. Benutzen Sie die Variable SEKTOR, so bricht der C64 mit einem Syntax-Error ab, da im Wort SEKTOR das BASIC-Befehlswort TO versteckt ist, was nicht erlaubt ist. Jede Fehlermeldung hat eine bestimmte Nummer, die hier in der Variablen NR abgespeichert wird. Sie können nun im Floppy-Handbuch nachschlagen, was diese Nummer bedeutet. In der Variablen FEHLER$ steht die Fehlermeldung in Kurzform. Die Variablen SPUR und SEKT geben an, auf welcher Spur bzw. auf welchem Sektor der Fehler auftrat. Auf diese Art und Weise lassen sich Fehler leicht bestimmen und somit auch beheben.

3.3.2 Auslesen des Fehlerkanals im Direktmodus

Haben Sie gerade ein Programm im Speicher, das Sie nicht abändern können, und tritt nun ein Floppyfehler auf, obwohl Sie nicht wissen warum, so muß der Fehlerkanal im Direktmodus ausgelesen werden. Geben Sie dazu folgendes ein:

```
OPEN 1,8,15
FORA=0TO39:POKE781,1:SYS65478:SYS65487:PRINTCHR$(PEEK(780));:SYS65
484:IF ST=0 THEN NEXTA
```

Nachdem Sie die RETURN-Taste gedrückt haben, erscheint die Fehlermeldung in folgender Form:

```
FEHLERNUMMER,FEHLER,TRACK,SEKTOR
```

Wollten Sie z.B. etwas laden, obwohl sich keine Diskette im Laufwerk befand, so erscheint:

```
XY,DRIVE NOT READY,00,00
```

Dieser Trick erfordert einige Erklärungen. Es ist von BASIC aus möglich, Maschinenspracheroutinen aufzurufen. Dies geschieht mit dem SYS-Befehl. Manche Routinen brauchen für ihre Arbeit bestimmte Werte, die in Registern abgelegt werden. Es gibt insgesamt vier Register:

Register	Adresse
Akkumulator	780
X-Register	781
Y-Register	782
Statusregister	783

Bei diesem Einzeiler wird eine Schleife höchstens 40mal durchlaufen, da eine Fehlermeldung höchstens 40 Zeichen lang ist. In der Schleife erhält erst das X-Register den Wert eins, der die Filenummer angibt. Daraufhin wird eine Routine namens CHKIN aufgerufen, die das aktuelle Eingabegerät (Tastatur) auf das im X-Register angegebene Gerät umschaltet. Im diesen Falle ist das die Floppy. Daraufhin wird ein Zeichen eingelesen, indem die Routine BASIN aufgerufen wird. Dieses Zeichen wird im Akkumulator (Adresse 780) abgelegt.

Der Print-Befehl gibt dieses Zeichen auf den Bildschirm aus. Anschließend wird die Tastatur als Eingabegerät und der Bildschirm als Ausgabegerät eingestellt. Danach wird das Status-Byte abgefragt. Ist es null, so wird die Schleife noch einmal durch-

laufen. Hat die Variable ST den Wert 64, so zeigt dies das Ende der Fehlermeldung an. Vergessen Sie nicht, nach diesen Befehlen den Kanal durch

```
CLOSE 1
```

wieder zu schließen, denn sonst würde beim eventuellen zweiten Öffnen ein File-Open-Error ausgegeben.

3.3.3 Ändern der Floppy-Adresse

Der C64 hat die Möglichkeit, 8 Floppys zu verwalten. Diese besetzen die Kanäle 8-15. Von Grund auf hat jede Floppy jedoch die Nummer 8. Wollen Sie diese ändern, so wird Ihnen dieses Programm dabei helfen:

```
10 OPEN 1,8,15
20 PRINT# 1,"M-W"CHR$(119);CHR$(0);CHR$(2);CHR$(NEUE_NUMMER +
32);CHR$(NEUE_NUMMER + 64)
30 CLOSE 1
```

Sie müssen hier lediglich für die Variable NEUE_NUMMER die von Ihnen gewünschte Geräteadresse angeben, z.B. 9. Wollen Sie eine Floppy mit einer anderen Gerätenummer als 8 'umpolen', so müssen Sie die Zeile 10 folgendermaßen ändern, wobei Sie für Nummer die alte Geräteadresse angeben müssen.

```
10 OPEN 1,NUMMER,15
```

Dieser Trick hat besonders dann Sinn, wenn Sie mehrere Floppys besitzen. Ein Nachteil läßt sich jedoch nicht abstreiten. Sobald Sie den Rechner ausschalten, werden alle Nummern wieder auf 8 gesetzt. Sie müssen also vor jeden neuem Arbeitsgang die Nummern neu setzen. Diesem Nachteil kann man durch das Durchtrennen zweier Drahtbrücken in der Floppy entgehen. Wie das funktioniert, können Sie in einschlägiger Literatur nachlesen.

3.3.4 Prüfung nach vorhandenem Schreibschutz

Nehmen wir einmal an, Sie wollen ein Programm schreiben, in dessen Verlauf Daten auf die Diskette geschrieben werden. Ihr Programm funktioniert wunderbar, doch wenn Sie vergessen haben, von Ihrer Datendiskette den Schreibschutz zu entfernen, gibt das Programm einen Fehler aus und bricht ab. Dies kann sehr ärgerlich sein, wenn es sich um wichtige Daten handelt. Es wäre also besser, wenn das Programm herausfinden könnte, ob die Diskette schreibgeschützt ist oder nicht. Das folgende Programm löst dieses Problem.

```
10 OPEN 1,8,15
20 PRINT# 1,"M-R";CHR$(0);CHR$(28)
30 GET# 1,SCHUTZ$
40 SCHUTZ = ASC(SCHUTZ$ + CHR$(0)) AND 16
50 IF SCHUTZ <> 0 THEN 100
60 PRINT "DIESE DISKETTE IST SCHREIBGESCHÜTZT, BITTE ENT-"
70 PRINT "FERNEN SIE ERST DEN SCHREIBSCHUTZ!!!"
80 GET A$ : IF A$ = "" THEN 80
90 PRINT "OK!"
100 REM hier wird das Programm fortgeführt
```

Das Programm öffnet den Floppy-Kanal und überprüft dann, ob die Diskette schreibgeschützt ist. Ist dies der Fall, so enthält die Variable SCHUTZ einen Wert ungleich 0, und der Benutzer wird aufgefordert, den Schreibschutz zu entfernen. Hat er dies getan, so soll er eine Taste drücken, und das Programm kann fortfahren. War die Diskette nicht schreibgeschützt, so verzweigt das Programm zu Zeile 100, in der das Hauptprogramm folgen soll. Diese Routine kann eingesetzt werden, wenn in Erfahrung gebracht werden muß, ob die Diskette schreibgeschützt ist oder nicht.

3.3.5 Floppy-Reset

Genau wie beim Computer, so läßt sich auch an der Floppy ein Reset ausführen. Dies ist z.B. dann sinnvoll, wenn ein Fehler auftrat und nun die Floppy-Lampe unentwegt blinkt. Soll nun

ein Floppy-Reset durchgeführt werden, so muß der Befehl "UJ" an die Floppy geschickt werden. Dies geschieht durch folgende drei Zeilen:

```
10 OPEN1,8,15
20 PRINT# 1,"UJ"
30 CLOSE 1
```

In Zeile 10 wird der Floppy-Kanal geöffnet, in Zeile 20 dann der Befehl "UJ" an die Floppy geschickt, und Zeile 30 schließt den Floppy-Kanal. Leider springt der Computer nach Beendigung des Programmes nicht zurück in die Eingabe-Warteschleife. Drücken Sie dann bitte

```
<RUN/STOP - RESTORE>.
```

Diese drei Zeilen lassen sich auch zu einer zusammenfassen, was den Vorteil hat, daß man diese Zeile im Direktmodus eingeben kann. Die Befehlsfolge sieht dann folgendermaßen aus:

```
OPEN1,8,15 : PRINT# 1,"UJ" : CLOSE1
```

oder einfach

```
OPEN1,8,15,"UJ" : CLOSE1
```

Sollte das einmal nicht seine Wirkung zeigen, so ist die Floppy "abgestürzt", und es hilft nur noch ein Aus- und Anschalten.

3.3.6 Schließen aller Kanäle

Haben Sie in einem Programm einmal mehrere Kanäle geöffnet, weil Sie beispielsweise mit der Floppy, dem Drucker und der Datasette gleichzeitig arbeiten wollen, so ist es sehr mühsam, diese Kanäle am Ende des Programms 'per Hand', also durch den Befehl

```
CLOSE Kanal
```

einzeln zu schließen. Ich möchte Ihnen in diesem Kapitel drei verschiedene Methoden zeigen, wie Sie dieses Problem elegant lösen können. Die erste Methode erinnert stark an das Schließen der Kanäle per Hand. Hier werden die CLOSE-Befehle jedoch nicht einzeln eingegeben, sondern in einer FOR-NEXT-Scheife abgearbeitet:

```
10 FOR KANAL = 1 TO 15
20 CLOSE KANAL
30 NEXT KANAL
```

Hierbei wird in einer Schleife, die 15mal durchlaufen wird - denn es gibt nur 15 Kanäle - jeweils ein Kanal geschlossen. Die zweite Methode besteht darin, einfach eine Routine ab Adresse 65511 mittels

```
SYS65511
```

aufzurufen, die jedoch nicht die Kanäle schließt, sondern sie lediglich vergißt. Dies bewirkt zwar das gleiche, ist jedoch nicht ganz 'sauber', und es kann evtl. zu späteren Störungen kommen. Die dritte und letzte Methode basiert darauf, daß der C64 sich 'merkt', welche Kanäle im Moment geöffnet sind. Außerdem ist in der Speicherzelle 152 die Anzahl der gerade geöffneten Kanäle gespeichert. Folgende Routine durchläuft eine FOR-NEXT-Schleife so oft, wie Kanäle geöffnet sind. Innerhalb dieser Schleife werden die einzelnen Kanäle geschlossen:

```
10 FOR KANAL = 1 TO PEEK (152)
20 CLOSE PEEK (601)
30 NEXT KANAL
```

3.3.7 Unscratch

Wußten Sie schon, daß ein Programm, das nach Eingabe von

```
OPEN 1,8,15,"S:PRO.NAME"
CLOSE 1
```

gelöscht wurde, noch gerettet werden kann? Geben Sie dazu einfach folgenden Befehl ein:

```
LOAD "*",8
```

Hier zeigt sich wieder der Vorteil von Wildcard-Zeichen, denn bei der Eingabe von

```
LOAD "PRG.NAME",8
```

hätte der C64 einen File-Not-Found-Error ausgegeben. Ein Wildcard-Zeichen ersetzt andere Zeichen. Das Sternchen steht stellvertretend für das zuletzt geladene Programm. Deshalb wird das eben gelöschte File noch einmal geladen. Das Laden von gelöschten Programmen setzt jedoch zweierlei voraus. Erstens darf nach dem ungewollten Löschen kein anderes Programm geladen werden, und zweitens darf nach dem Löschen kein Floppy-Reset (siehe Kapitel 3.3.5) durchgeführt werden.

3.3.8 Formatierung in einer Sekunde

Haben Sie eine volle Diskette und brauchen Sie die Daten nicht mehr, so ist es sinnvoll, diese Diskette neu zu formatieren, denn das einzelne Löschen der Programme ist recht zeitaufwendig und kompliziert. Der Formatierungsvorgang dauert jedoch ca. eine Minute, was bei mehreren Disketten schon recht nervend sein kann. Doch diese Zeit läßt sich stark verkürzen. Legen Sie eine zu löschende Diskette ins Laufwerk, und geben Sie folgendes ein (Achtung: Die Daten werden gelöscht!):

```
OPEN 1,8,15,"N:TIPS&TRICKS"
CLOSE 1
```

Das Diskettenlaufwerk läuft kurz an, doch mehr passiert auch nicht. Geben Sie nun einmal

```
LOAD "$",8
LIST
```

ein, und Sie sehen, daß die Diskette nun leer ist. Sie haben wieder 664 Blöcke frei. Vielleicht ist Ihnen bei der Eingabe der kleine - aber feine - Unterschied zwischen dem normalen Formatierungsbefehl und dem abgeänderten gar nicht aufgefallen. Die ID-Angabe wurde weggelassen, was bewirkt, daß lediglich das Inhaltsverzeichnis der Diskette, nicht aber die Diskette selbst gelöscht wird. Es werden also die Daten, die sich auf der Diskette befinden, nicht gelöscht, sondern nur 'vergessen'.

Sollten Sie geheime Daten auf Ihren Disketten haben, die Sie nun nicht mehr benötigen, so sollten Sie diesen Trick nicht anwenden, da man sich die Daten noch mit einem Diskettenmonitor angucken kann. Also: Im Zweifelsfalle immer den normalen Formatierungsbefehl benutzen!

3.3.9 1328 Blocks Free

Sind Sie Besitzer einer Floppy 1571 und eines C64/C128? Haben Sie sich schon immmer geärgert, daß Sie - obwohl Ihr Laufwerk zwei Schreib-/Leseköpfe hat - Sie Ihre Disketten nur einseitig verwenden können? Dann sollten Sie nun dieses Kapitel lesen, denn danach ist Schluß mit einseitigen Disketten. Wie Sie sich schon denken können, ist es möglich, mit der 1571 im C64-Modus die Diskette beidseitig zu formatieren und auch zu benutzen. Nehmen Sie bitte eine neue Diskette oder eine, die unbenötigte Daten enthält, denn wir wollen diese nun zweiseitig formatieren. Geben Sie dazu folgendes ein:

```
OPEN 1,8,15,"U0>M1"
PRINT# 1,"N:TIPS&TRICKS,AP"
CLOSE 1
```

Sollten Sie keine Diskette im Laufwerk haben, so gibt die Floppy nach dem ersten Befehl einen Fehler aus, den Sie aber ignorieren können. Spätestens vor der Eingabe des zweiten Befehles sollten Sie die zu formatierende Diskette ins Diskettenlaufwerk legen. Das Laufwerk läuft nun an. Nach einiger Zeit hat es sich wieder beruhigt. Wenn Sie nun einmal das Directory laden, so wird die Meldung

```
1328 BLOCKS FREE
```

ausgegeben. Sie haben nun doppelt soviel Speicherplatz wie vorher zur Verfügung. Der Trick ist eigentlich recht einfach: Mit dem ersten Befehl wird der Modus der 1571 umgeschaltet. Die 1571 hat nämlich einen C64- und einen C128-Modus. Arbeiten Sie im C64-Modus, so schaltet auch die Floppy automatisch in den C64-Modus. Man muß nun nur den C128-Modus der 1571 einschalten, und schon arbeitet das Diskettenlaufwerk zweiseitig. Das Umschalten in den C128-Modus geschieht durch die erste Zeile. Danach wird die Diskette durch den NEW-Befehl ganz normal zweiseitig formatiert, und der dritte Befehl schließt den Kanal 1.

3.4 Sonstiges zur Floppy

Hier nun noch zwei kleine Tricks, die in keines der drei vorherigen Kapitel paßten, weshalb sie jedoch nicht minder nützlich sind. Hier sind sie!

3.4.1 Löschen des Komma-Files

Ihnen ist sicherlich schon einmal ein File namens "," im Directory aufgefallen, das dadurch entstanden ist, daß ein Programm, das mit der Diskette arbeitet, einen kleinen Fehler hatte. Nun ist es jedoch nicht schön, immer ein unerwünschtes File auf der Diskette zu haben. Versucht man, dieses File mittels

```
OPEN 1,8,15,"S:,"
CLOSE 1
```

zu löschen, so wird man sein blaues Wunder erleben, denn das File verschwindet nicht. Kommt man nun auf die Idee, dieses File mittels

```
OPEN 1,8,15,"R:NEUER_NAME=,"
CLOSE 1
```

umzubenennen, so wird man leider auch hier enttäuscht. Es gibt jedoch einen Weg, solch ein Komma-File zu entfernen, der mitunter recht aufwendig sein kann. Benennen Sie einfach alle Files um, die einen Namen von nur einem Buchstaben Länge haben. Dies geschieht mit folgendem Befehl:

```
OPEN 1,8,15,"R:NEUER_NAME=ALTER_NAME"
CLOSE 1
```

Hier müssen Sie natürlich noch die Variablen NEUER_NAME und ALTER_NAME durch die entsprechenden Namen ersetzen. Haben Sie dies getan, so sollte sich nur noch ein File auf der Diskette befinden, dessen Name eine Länge von einem Buchstaben hat. Überzeugen Sie sich davon, denn sonst können wichtige Daten verlorengehen. Geben Sie nun folgende Befehle ein:

```
OPEN 1,8,15,"S:?"
CLOSE 1
```

Das File mit dem Namen "," ist nun verschwunden, denn das Fragezeichen ist auch ein Wildcard-Zeichen. Das Fragezeichen steht stellvertretend für jedes Programm, das eine Länge von einem Buchstaben hat. Da es nur das Komma-File gab, wurde dieses also gelöscht. Dies ist auch der Grund, warum vorher alle Programme, deren Name ebenfalls nur einen Buchstaben umfaßte, umbenannt werden mußten, denn sonst wären sie auch alle gelöscht worden. Übrigens lassen sich durch das Sternchen als Wildcard-Zeichen und dem SCRATCH-Befehl alle Programme auf der Diskette löschen (probieren Sie das jedoch jetzt nicht aus!):

```
OPEN 1,8,15,"S:*"
CLOSE 1
```

3.4.2 Verkürzen der Floppy-Zugriffszeit

Es ist möglich, die Zugriffszeit der Floppy zu verkürzen. Unter Floppy-Zugriffszeit versteht man die Zeit, die der Schreib-/Lesekopf braucht, um eine bestimmte Stelle auf der Diskette zu erreichen und dann auf die Daten 'zugreifen' zu können. Wäh-

rend sich der Schreib-/Lesekopf zu einer Stelle auf der Diskette bewegt, werden mehrere Interrupts ausgelöst, in denen verschiedene Zustände überprüft werden, die uns hier nicht weiter interessieren sollen. Da dies jedoch kostbare Zeit kostet und zu diesem Zeitpunkt unwichtig ist, ist es sinnvoll, diese Interrupts (deutsch: Unterbrechungen) zu unterbinden. Dies geschieht durch folgende Befehle:

```
OPEN 1,8,15
PRINT# 1,"M-W"+CHR$(7)+CHR$(28)+CHR$(1)+CHR$(15)
CLOSE 1
```

Wenn Sie nun durch den Befehl

```
LOAD "$",8
```

das Directory der Diskette laden oder durch

```
LOAD "PRG.NAME",8
```

ein Programm laden, so werden Sie feststellen, daß die Meldung

```
LOADING $
```

bzw. die Meldung

```
LOAD "PRG.NAME"
```

wesentlich schneller auf dem Bildschirm erscheint. Dieser Trick ist wieder ein gutes Beispiel dafür, wie effektiv doch ein einfacher Befehl sein kann.

[illegible] und über der Kernel-[illegible] zu einer Stelle [illegible] werden mehrere Interrupts ausgelöst, in denen verschiedene Zeichen überprüft werden, die uns hier nicht weiter interessieren sollen. Da dies jedoch kostbare Zeit kostet und in diesem Zeitpunkt unwichtig ist, ist es sinnvoll, diese Interrupts (deutsch: Unterbrechungen) zu unterbinden. Dies geschieht durch folgende Befehle:

```
[illegible]
[illegible]
[illegible]
```

[illegible] Sie nun [illegible] den Befehl

```
[illegible]
```

[illegible] Diskette laden oder durch

```
[illegible]
```

ein Programm [illegible] laden, so werden Sie feststellen, daß die Meldung

```
[illegible]
```

bzw. die Meldung

```
[illegible]
```

wesentlich schneller auf dem Bildschirm erscheint. Dieser Trick ist somit ein gutes Beispiel dafür, wie effektiv doch ein kleiner Trick sein kann.

4. BASIC

'Warum ein eigenes Kapitel über BASIC?', werden Sie sagen. Dazu hat doch jeder C64-Besitzer sein Handbuch. Ich will Ihnen zeigen, wie man, ohne auf Maschinensprache zurückgreifen zu müssen, das Letzte aus dem zugegebenermaßen knapp gehaltenen BASIC herausholen kann.

4.1 Geschwindigkeit

Der erste Nachteil des BASIC V2 ist wohl die geringe Ausführungsgeschwindigkeit. Betrachten wir zunächst einmal die prinzipielle Funktionsweise der BASIC-Interpreters: Normalerweise 'versteht' der C64 gar kein BASIC, hardwaremäßig ist er nur in der Lage, Maschinensprachebefehle auszuführen. Diese bestehen nur aus Zahlen, und es gibt nur ganz elementare Befehle wie AND, OR, Verschieben, Plus und Minus, nicht einmal eine Multiplikation ist möglich. Allerdings bieten moderne symbolische Makro-Assembler hier viele Hilfen an, so daß es mit ein wenig Übung praktisch für jeden möglich sein sollte, Programme in Assembler zu schreiben (als Assembler bezeichnet man Hilfsprogramme, die zur Erstellung von Maschinensprache-programmen dienen).

Das BASIC des C64 ist nun so aufgebaut, daß bei der Ausführung jeder Befehl analysiert wird und eine entsprechende Betriebssystemroutine im ROM aufgerufen wird. Obwohl diese natürlich in Maschinensprache geschrieben sind, ist BASIC sehr langsam, weil die Analyse eine Menge Zeit in Anspruch nimmt. Eine Hilfe bieten hier sogenannte Compiler, wie z.B. das Programm BASIC 64 von DATA BECKER. Diese Programme übersetzen BASIC-Programme in Maschinensprache, beschleunigen diese also. An "echte" Maschinenprogramme kommen sie in der Geschwindigkeit allerdings nicht heran, da die Hardware, die Möglichkeiten des Prozessors, Daten in Registern zu speichern, und andere Punkte nie ganz effektiv ausgenutzt werden können.

Auch wenn es sehr problematisch ist, einen Zeitfaktor anzugeben, da verschiedene BASIC-Befehle unterschiedlich stark beschleunigt werden, kann man ganz grob sagen, daß ein durchschnittlicher Compiler BASIC-Programme etwa um das zehnfache beschleunigt. Verlassen Sie sich aber beim Kauf auf keinen Fall auf die Herstellerangabe, sondern informieren Sie sich anhand sogenannter "Benchmark-Tests", die häufig in Fachzeitschriften veröffentlicht werden. Diese Tests sind kleine Programme, die spezielle Befehlsgruppen benutzen. Die Ausführungszeit dieser Tests gibt Aufschluß über die Qualität des Compilers.

Nur zur Vollständigkeit sei erwähnt, daß es neben Maschinensprache und BASIC (Interpreter und Compiler) inzwischen eine ganze Reihe weiterer Programmiersprachen für den C64 gibt. Auch hierbei gilt das oben Gesagte, es gibt sogar Sprachen (wie Pascal oder C), die nur als Compiler geliefert werden, d.h. Sie müssen Ihr Programm komplett schreiben, übersetzen lassen (der Übersetzungsvorgang kann einige Minuten in Anspruch nehmen), und dann testen. Bei einem Fehler landen Sie nicht wie von BASIC gewohnt mit einem Syntax-Error im Editor, sondern entweder mit einer Systemfehlermeldung im Hauptmenü der Sprache, im BASIC oder sonstwo, oder das System stürzt ganz ab.

Aber mit einigen einfachen Tricks kann man auch in BASIC noch eine Menge optimieren. Zur Zeitmessung werden wir die interne Uhr TI$ benutzen. Durch TI$="000000" wird diese gestartet, in der Systemvariablen TI steht die seit dem Start verstrichene Zeit in 1/60 Sekunden.

4.1.1 Punkte statt Null

Wenn Sie in einem Programm eine Null zu Berechnungen oder Vergleichen benutzen, so geben Sie statt der Null einen einfachen Punkt (.) ein. Der BASIC-Interpreter speichert nämlich Zahlen bei der Eingabe intern als Strings, also als Zeichenketten, ab, die erst bei der Ausführung in Fließkommazahlen übersetzt

werden. Der Punkt wird nun schneller eingelesen als eine Null, die erst in eine Zahl übersetzt werden muß. Hierzu zwei Beispiele:

```
10 TI$="000000"          10 TI$="000000"
20 FOR T=1 TO 1000       20 FOR T=1 TO 1000
30 A=.                   30 A=0
40 NEXT T                40 NEXT T
50 PRINT TI              50 PRINT TI
```

Ausgabe:

```
152                      188
```

Entsprechendes gilt natürlich auch für Zahlen wie 0.5, welche als .5 eingegeben werden sollten (wie sie übrigens im PRINT-Befehl auch ausgegeben werden).

4.1.2 Variablen statt Konstanten

Auf einer ähnlichen Grundlage beruht folgender Tip: Konstanten, die häufig benötigt werden, sollten am Anfang des Programmes als Variablen abgespeichert werden. So durchlaufen diese die erwähnte String-Umwandlungsroutine nur einmal, und das Programm wird dementsprechend schneller. Beispiel:

```
10 TI$="000000":KO=17.23456     10 TI$="000000"
20 FOR T=1 TO 1000              20 FOR T=1 TO 1000
30 A=KO                         30 A=17.23456
40 NEXT T                       40 NEXT T
50 PRINT TI                     50 PRINT TI
```

Ausgabe:

```
172                             1505
```

4.1.3 Fließkommavariablen statt Integer

Nachdem es nun offensichtlich Zeit spart, Variablen statt Konstanten zu benutzen, kommt es nun noch dicker: Benutzen Sie Fließkommavariablen statt Integer. Ja, das ist kein Druckfehler, auch wenn man Ihnen bei anderen Sprachen ziemlich genau das Gegenteil erzählt haben mag (und das zu Recht).

Der Hinweis beruht auf der Eigenart des BASIC-Interpreters, daß dieser nicht auf Integer-Rechenoperationen zurückgreift, sondern alle Zahlen, auch wenn sie mit %-Zeichen als Integer definiert wurden, vor der Weiterverarbeitung in Fließkommazahlen umwandelt.

Auch Speicherplatz sparen diese Variablen nicht, jede Variable, ob Integer oder Floating-Point, nimmt grundsätzlich 7 Bytes ein, wobei die ersten beiden für den Variablennamen stehen und bei den Integer-Variablen die letzten 3 Variablen einfach unbelegt in der (Speicher-) Landschaft herumstehen. Die einzige Ausnahme hiervon sind Arrays, hier sind Integer-Arrays speichersparender als Fließkomma-Arrays. Beispiel:

```
10 TI$="000000":KO=7        10 TI$="000000":KO%=7
20 FOR T=1 TO 1000          20 FOR T=1 TO 1000
30 A=KO                     30 A=KO%
40 NEXT T                   40 NEXT T
50 PRINT TI                 50 PRINT TI
```

Ausgabe:

```
171                         187
```

Bitte beachten Sie, daß viele BASIC-Compiler (u.a. BASIC 64) im Gegensatz zum Interpreter über Integer-Routinen verfügen, d.h., hier sollten Integer-Variablen so häufig wie möglich benutzt werden.

4.1.4 Unterprogramme an den Anfang

Der nächste Tip beruht auf der Tatsache, daß das Commodore-BASIC bei Aufrufen von Unterprogrammen und Goto-Sprüngen nicht die tatsächliche Adresse der Zeile abspeichert, sondern erst bei der Ausführung das Programm von vorne bis hinten nach der angegebenen Zeilennummer durchsucht. Eine Menge Zeit können Sie vor allem bei längeren Programmen sparen, indem Sie oft benötigte Unterprogramme nicht ans Ende, sondern ganz an den Anfang des Programmes setzen. Wegen der Kürze des Beispielprogrammes macht sich der Effekt hier nicht so sehr bemerkbar, bei längeren Programmen kann man so aber ganz erheblich Zeit sparen. Beispiel:

```
10 GOTO 30                  10 TI$="000000"
20 RETURN                   20 FOR T=1 TO 1000
30 TI$="000000"             30 GOSUB 60
40 FOR T=1 TO 1000          40 NEXT T
50 GOSUB 20                 50 PRINT TI:END
60 NEXT T                   60 RETURN
70 PRINT TI
```

Ausgabe:

```
159                         164
```

4.1.5 FOR-NEXT statt IF THEN GOTO

Falls Sie noch, weil Sie es etwa von anderen Rechnern gewohnt sind, Schleifen mit IF THEN GOTO konstruieren, sollten Sie sich dies schleunigst abgewöhnen, die vom BASIC V2 bereitgestellte FOR-NEXT-Schleife ist bedeutend schneller.

```
10 TI$="000000"             10 TI$="000000"
20 FOR T=1 TO 1000          20 T=1
30 NEXT T                   30 T=T+1
40 PRINT TI                 40 IF T<1000 THEN 30
                            50 PRINT TI
```

Ausgabe:

```
81                          560
```

4.1.6 Multiplizieren statt Potenzieren

Viele Rechenroutinen des BASIC sind recht uneffektiv, so wird z.B. die Potenz a hoch 2 nach der Formel

$$a^2 = e^{(2*\ln a)}$$

berechnet. Mathematisch ist das zwar korrekt, aber man kann sich vorstellen, wie lange es dauert, bis diese Routine durchlaufen ist. Statt a^2 sollten Sie also künftig in Ihren Programmen einfach a*a, statt a^3 a*a*a schreiben. Übrigens ist das auch genauer, da nicht so viele Berechnungen angestellt werden und sich Rundungsfehler nicht so sehr fortpflanzen.

```
10 TI$="000000"          10 TI$="000000"
20 FOR T=1 TO 1000       20 FOR T=1 TO 1000
30 A=9*9                 30 A=9^2
40 NEXT T                40 NEXT T
50 PRINT TI,A            50 PRINT TI,A
```

Ausgabe:

```
324  81                  3363  81.0000001
```

4.1.7 Rundungsfehler

Wie Sie im letzten Beispiel gesehen haben, war die linke Variante nicht nur schneller, sondern auch genauer. Die im rechten Beispiel demonstrierten Rundungsfehler treten übrigens bei diversen Funktionen auf, nicht nur bei der Potenzfunktion, der Programmierer sollte sich dessen stets bewußt sein. Probieren Sie einmal folgendes Programm aus:

```
10 A=0
20 A=A+0.1
30 IF A=1 THEN END
40 GOTO 20
```

Man sollte erwarten, daß die Variable a von 0 in 1/10-Schritten nach 1 hochgezählt wird und dann das Programm endet. Auf dieses Ende allerdings können Sie lange warten, durch Rundungsfehler in der Additionsroutine wird die Zahl nie 1, und es wird endlos hochgezählt. Das Tückische daran ist, daß die Zahl zur Bildschirmausgabe noch einmal gerundet wird, so daß der Fehler auf Anhieb gar nicht zu sehen ist, wie Sie durch Einfügen der Zeile

```
25 PRINT A
```

feststellen können. Erst nach ca. 35 Additionen hat sich der Fehler so weit fortgepflanzt, daß er auch auf dem Bildschirm zu sehen ist. Eine wirkliche Lösung des Problems ist nur das Ersetzen der Zeile 30 durch

```
30 IF A+.05>=1 THEN END
```

oder besser durch eine For-Next Schleife. Wenn Sie programmieren, sollten Sie immer an diese Eigenschaft des Commodore-BASIC denken, um Programmfehler zu vermeiden, die sehr schwer zu lokalisieren sind. Übrigens ist es auch schneller, eine Multiplikation mit 2 durch eine Addition zu ersetzen.

4.1.8 IF-THEN-Abfragen

Wenn in Abhängigkeit von mehreren Bedingungen eine Befehlsfolge aufgerufen werden soll, bietet sich eine Verzweigung nach dem Schema

```
IF A AND B THEN befehle
```

an. Wenn a und b oft wahr sind, so ist dieses wohl auch die schnellste Variante. Es kann jedoch auch sein, daß einer der beiden Teilwahrheitswerte voraussichtlich während der Laufzeit so gut wie immer falsch ist (z.B. in einem Unterprogramm, das einen Bildschirmpunkt setzt und abfragt, ob die Koordinaten innerhalb der erlaubten Grenzen liegen, was wohl normalerweise

der Fall ist. In diesem Fall kann Zeit gespart werden, indem die Abfrage "IF A AND B THEN befehle" durch "IF A THEN IF B THEN befehle" ersetzt wird, wobei a nur selten wahr ist.

```
10 A=1:B=1:TI$="000000"
20 B=B+1
30 IF B=110 THEN B=0:
   A=A+1
40 IF A>10 THEN IF
   B>100 THEN PRINT TI:END
50 GOTO 20
```

```
10 A=1:B=1:TI$="000000"
20 B=B+1
30 IF B=110 THEN B=0:
   A=A+1
40 IF A>10 AND B>100
   THEN PRINT TI:END
50 GOTO 20
```

Ausgabe

```
980                              1312
```

Entsprechend kann man auch ein

```
IF A OR B THEN BEFEHLE
```

durch folgende Befehlszeilen ersetzt werden, wenn A fast immer wahr ist.

```
10 IF NOT A THEN IF NOT B THEN 30
20 BEFEHLE
30 REM hier geht es weiter
```

4.1.9 Viele Befehle in einer Zeile

Wenn mehrere Befehle in eine Zeile gesetzt statt über mehrere Zeilen verteilt werden, so spart auch dies ein wenig Zeit. Übrigens kann es auch übersichtlicher sein, nämlich dann, wenn durch die Aufsplittung in zu viele Einzelzeilen logisch zusammenhängende Blöcke nicht mehr als solche erkannt werden können.

Allerdings gilt auch hier: Zuviel des Guten ist wiederum schlecht. Werden alle Zeilen grundsätzlich mit 80 Zeichen ausgenutzt, so verliert man in dem Befehlswirrwar schnell den Überblick.

```
10 TI$="000000"
20 FOR T=1 TO 500
30 A=1:B=1:C=1:E=1:F=1:G=1
40 NEXT T
50 PRINT T
```

```
10 TI$="000000"
20 FOR T=1 TO 500
30 A=1
40 B=1
50 C=1
60 D=1
70 E=1
80 F=1
90 G=1
100 NEXT T
110 PRINT TI
```

Ausgabe:

```
432                    442
```

Grundsätzliches zu optimierenden Maßnahmen

Sie sollten ein Gefühl dafür entwickeln, wann es sinnvoll ist, zeitsparende Maßnahmen zu ergreifen, da diese häufig andere Nachteile mit sich bringen. So ist es zweifellos zeitsparender, möglichst viele Befehle in eine Zeile zu "packen". Allerdings ist es übersichtlicher, immer nur logisch zusammenhängende Befehle in eine Zeile zu schreiben und für einen neuen logischen Befehlsblock eine neue Zeile anzufangen. Gerade die Unübersichtlichkeit trägt dazu bei, daß eventuelle Programmierfehler (die sich wohl nicht vermeiden lassen) schwerer aufzufinden und zu beheben sind.

Sie sollten sich darüber bewußt sein, daß eine solche Maßnahme Zeiten im Millisekundenbereich einspart, die ein normaler Anwender ohnehin nicht bemerkt, wenn die entsprechende Befehlsfolge nicht in einer Schleife liegt, die häufig durchlaufen wird.

Das Kriterium für geschwindigkeitsoptimierende Maßnahmen muß also stets lauten: Wie oft wird die fragliche Befehlssequenz voraussichtlich durchlaufen? Wird also viel Zeit durch Optimierung eingespart?

4.2 Editor

Gegenüber anderen zeilenorientierten Editoren, mit denen man ein Programm zwar auflisten kann, aber dann nicht einfach mit dem Cursor eine Zeile hochgehen kann, um dort mal eben etwas zu ändern, sondern einen Befehl "Edit [Zeilennummer]" benötigt, um eine Zeile zu ändern, ist der Commodore-Editor zwar ein Fortschritt, aber objektiv betrachtet auch nicht "das Gelbe vom Ei" Einige Nachteile kann man jedoch ganz gut wettmachen:

4.2.1 Einrücken

Von anderen Sprachen kennt man die Möglichkeit, bei komplizierten Algorithmen und verschachtelten Strukturen Zeilen einzurücken. Ein Beispiel dafür wären Befehle, die innerhalb einer FOR-NEXT-Schleife stehen, so daß man den Programmablauf sofort erkennt. Der BASIC-Editor hat nun aber leider die unangenehme Eigenschaft, diese Leerzeichen sofort wieder zu löschen, was wohl ursprünglich den durchaus ehrenwerten Zweck hatte, die Programmgeschwindigkeit zu erhöhen, doch wurde dies nicht konsequent weitergedacht, denn zwischen zwei Befehlen werden diese Leerzeichen nicht gelöscht.

Diese Tatsache führt zu einem besonders simplen, aber wirkungsvollen Tip: Leiten Sie die einzufügenden Zeilen mit einem Doppelpunkt ein, und nun können so viele Leerzeichen folgen, wie Sie es für sinnvoll erachten.

```
10 FOR X=0 TO 39
20 : FOR Y=0 TO 24
30 : : POKE 1024+X+Y*40,1
40 : NEXT Y
50 NEXT X
```

Mit diesem Programm können Sie übrigens prüfen, ob Sie über ein altes oder ein neues Betriebssystem verfügen, bei älteren Betriebssystemversionen werden die bereits auf dem Bildschirm stehenden Zeichen durch 'A's ersetzt, in neueren Versionen wird der ganze Bildschirm mit 'A' gefüllt.

4.2.2 Entfernen der Einrückungen

Nun sind diese Doppelpunkte bei der Erstellung eines Programmes ganz praktisch, machen aber die oben beschriebenen Geschwindigkeitsoptimierungen wieder zunichte. Es gibt jedoch eine Methode, diese Leerzeichen wieder sicher zu entfernen. Durch den Poke

```
Poke 129,58
```

gelangt der C64 in einen Modus, indem er jedes nicht in Anführungszeichen stehende Leerzeichen mit einem Error bemäkelt, so daß Sie, - nachdem Sie die Einrückungen von Hand entfernt haben - hiermit testen können, ob Sie nichts vergessen haben. Mit

```
Poke 129,32
```

wird dieser Modus wieder aufgehoben.

4.2.3 REM-Killer

REM-Befehle sind eine sehr praktische Einrichtung, um ein Programm zu dokumentieren, damit es auch später noch durchschaut oder von anderen geändert werden kann. Ein REM macht sich z.B. bei Unterprogrammsprüngen sehr gut, wenn dahinter steht, was das Unterprogramm macht. Komplizierte Ausdrücke können erläutert, Variablen erklärt werden o.ä.

Allerdings ist ein REM-Befehl wieder ein Hemmschuh in Sachen Ausführungsgeschwindigkeit und Speicherverbrauch. Als Faustregel sollten Sie beachten, REMs möglichst außerhalb von Schleifen zu positionieren, da die REM-Befehle dann weniger oft durchlaufen werden. Mit folgendem Hilfsprogramm können Sie die am Anfang einer Zeile stehenden REMs aus Ihrem Programm automatisch entfernen lassen. Es ist also zweckmäßig, zunächst ein Programm zu schreiben,zu testen und REMs zu setzen (möglichst an Zeilenanfänge). Sobald das Programm funktioniert, können Sie es mit diesem REM-Killer bearbeiten.

Wenn Sie irgendwann einmal das Programm ändern wollen, benutzen Sie die Version mit REMs, wenn Sie es "nur" benutzen wollen, diejenige ohne REMs. Dieser Hinweis gilt natürlich auch für die oben beschriebenen Einrückungen. Bitte beachten Sie, daß Sie keine Zeilennummern, die größer als 63000 sind, benutzen sollten, um nicht in Konflikt mit dem REM-Killer zu kommen. Sie können den REM-Killer einfach mit Ihrem Programm zusammen eintippen (lang ist er ja schließlich nicht) oder aber einmal eintippen und künfig mit dem unter Kapitel 4.2.6 beschriebenen Merge anhängen. Hier nun das Listing:

```
63000 AD=2049
63010 NA=PEEK(AD)+PEEK(AD+1)*256
63020 IF PEEK(AD+4)=143 GOTO 63100
63030 AD=NA: IF PEEK(AD)<>0 GOTO 63010
63040 END
63100 POKE 251,AD AND 255:POKE 252,AD/256
63110 PRINT "<HOME>"PEEK(AD+2)+PEEK(AD+3)*256:PRINT "GOTO 63130"
63120 POKE 198,3:POKE 631,19:POKE 632,13:POKE 633,13:END
63130 NA=PEEK(251)+PEEK(252)*256:GOTO 63030
```

Aufgerufen wird das Programm übrigens mit "goto 63000". Um die Funktionsweise des Programmes zu verstehen, muß man wissen, wie BASIC-Programme intern abgespeichert werden. Wenn ein BASIC-Befehlswort eingegeben wird, so wird nicht dieses Wort als solches abgespeichert, das würde zuviel Speicher verbrauchen und in der Ausführung zu langsam sein.

Statt dessen wird für jeden Befehl eine Kennziffer, ein sogenanntes Token, im Speicher abgelegt. An dieser erkennt der Interpreter bei der Ausführung, welche Betriebssystemroutine des ROM er aufrufen soll, und an dieser erkennt die List-Routine, welches Wort sie anzeigen soll. Das Token für REM ist 143, eine komplette Token-Liste befindet sich im Anhang. Experimentieren Sie doch einmal mit dem Programm und lassen z.B. alle DATA-Zeilen löschen. Im Speicher liegt ein BASIC-Programm nun wie folgt:

Zeilen-nummer 1. Zeile	Adresse der nächsten	Token 1	evt. Parameter	Token 2	...	Zeilen-nummer 2. Zeile
1.+2.Byte	3.+4.Byte	5.Byte	...	...		

Abb. 2 BASIC-Zeile

Die erste Zeile steht normalerweise ab 2049 im Speicher, durch geeignete Pokes kann man dies aber ändern. (siehe Ausführungen bei "Anhängen"). So, nun wissen wir, wie man alle REM-Zeilen finden kann. Es besteht jetzt aber noch das Problem, wie man aus einem Programm heraus eine Zeile löschen kann, ohne daß das Programm sich selbst zerstört (schließlich stimmt nach dem Löschen einer Zeile z.B. der Programmzähler nicht mehr).

Die Lösung ist ebenso einfach wie wirkungsvoll: Die Zeile wird gar nicht vom Programm heraus gelöscht, sondern im Direkt-Modus. Nun bekommen Sie aber nicht etwa alle REM-Zeilen angezeigt, um diese zu löschen, auch hier wird wieder eine sehr nützliche Eigenart des C64 ausgenutzt. Wenn eine Taste gedrückt wird, so schreibt das Betriebssystem diese in einen Zwischenspeicher, den sogenannten Tastaturpuffer. Dieser Tastaturpuffer liegt zwischen den Adressen 631 und 641. In der Speicherstelle 198 steht immer, wie viele Zeichen zur Zeit im Tastaturpuffer zwischengespeichert sind.

Wird nun eine Taste gedrückt, so liest das Betriebssystem den Inhalt der Speicherstelle 198 aus. Wenn dieser 10 oder größer ist, wird der Tastendruck ignoriert. Andernfalls wird der ASCII-Code der Taste nach 631+peek(198) geschrieben und der Inhalt der Speicherstelle 198 um eins erhöht. Beim Lesen eines Zeichens von der Tastatur geht das Betriebssystem genau umgekehrt vor. Ist der Inhalt von 198 0, so wurde keine Taste gedrückt. Andernfalls wird peek(peek(198)+630) als gedrückte Taste angenommen und der Inhalt von 198 um eins heruntergezählt.

Dieser Routine bedienen sich alle C64-Funktionen, die die Tastatur benutzen also auch der Direkt-Modus. Der Rest ist nun fast trivial: Man schreibt die zu löschende Zeilennummer ganz oben auf den Bildschirm, darunter ein GOTO auf das eigene Programm, um danach weitermachen zu können, schreibt in den Tastaturpuffer die ASCII-Werte von Home und zweimal Return, und man beendet das Programm. In der Tat wird die Zeile gelöscht und in das Programm zurückgekehrt.

Einen Schönheitsfehler hat unser Verfahren jetzt noch: Beim Löschen einer Zeile werden alle Variablen gelöscht, was bedeutet, daß sich auch unser Zeiger AD in Wohlgefallen auflöst. Lösung: Vorher wird dieser, fein säuberlich nach Low- und Highbyte getrennt, in die freien Speicherstellen 251 und 252 gepoket und hinterher dort wieder hergeholt, d.h. nicht er, sondern der Zeiger NA, da nach dem Löschen der Zeile natürlich die nächste Zeile da anfängt, wo vorher die gelöschte Zeile anfing.

4.2.4 Zeilenlöschroutine

Leider fehlt beim C64 ein Befehl, der dazu dient, eine Reihe von aufeinanderfolgenden BASIC-Zeilen zu löschen. Ich möchte Ihnen hier eine kurze Routine vorstellen, diesen Befehl zu simulieren:

```
63000 POKE 214,22:POKE 211,0:SYS 58640:PRINT I
63010 PRINT "I=";I+1;":J=";J;": IF I<J-1 THEN 60000
      <4XCURSOR UP>":POKE 198,2:POKE 631,13:POKE 632,13
```

Die Funktionsweise ist im Prinzip dieselbe wie beim REM-Killer, nur daß hier die Variablen nicht in Speicherstellen zwischengespeichert werden, sondern einfach auf dem Bildschirm angezeigt und durch das Return wieder eingelesen werden. Die Zeilen werden gelöscht, indem nacheinander alle Zeilennummern auf dem Bildschirm angezeigt werden, um mit dem Tastaturpuffer-Return übernommen zu werden. Angenommen, Sie wollen alle Zeilen zwischen 1000 und 1200 löschen. Geben Sie dazu einfach ein

```
I=1000:J=1200:goto 63000
```

und schon werden durch dieses nur zwei Zeilen lange Programm die entsprechendenen Zeilen gelöscht.

4.2.5 Auto-Line

Ebenfalls fehlt dem C64 ein Befehl zur automatischen Vorgabe der Zeilennummern. Auch hier gehen wir wieder nach demselben Schema vor:

```
63000 POKE 214,22:POKE 211,0:SYS 58640:PRINT I;":"
63010 PRINT "I=";I+S;":J=";J;":S=";S;": IF I<J-1 THEN 60000
      <4XCURSOR UP>"
63020 POKE 198,2:POKE 631,13:POKE 632,13
```

Nun geben Sie, wie vom obigen Utility schon gewohnt, die Parameter über Variablen vor und springen in die Routine. Beispiel:

```
I=10:J=1000:S=10:goto 63000
```

Hierbei bedeutet

I erste Zeile
J letzte Zeile
S Schrittweite

Nach dem Start des Programmes wird ein vollständig durchnumeriertes Leerprogramm erzeugt, das nur aus den Zeilennummern und einem Doppelpunkt besteht. Dieses Rumpfprogramm kann dann von Ihnen überschreiben kann.

4.2.6 Merge

Dieser Befehl ist wohl für einen Editor geradezu essentiell. Nehmen wir an, Sie haben ein Unterprogramm zur Bildschirmausgabe geschrieben und wollen dies in verschiedenen Programmen nutzen. Wenn es nach Commodore ginge, müßten Sie dieses

Unterprogramm in jedem Programm neu eintippen. Wenn Sie das Unterprogramm mit sehr hohen Zeilennummern versehen haben, die höher als alle bis dato in Ihrem Programm benutzten Zeilennummern sind, so besteht überhaupt kein Problem. In der sogenannten ZEROPAGE, den untersten 256 Bytes des C64, befinden sich viele wichtige Speicherstellen, unter anderem die Zeiger auf BASIC-Programme.

Dabei bedeuten die einzelnen Zeiger folgendes:

43-44	BASIC-Speicher-Anfang
45-46	BASIC-Programmende, Variablenanfang
47-48	Anfang des Array-Speichers
49-50	Ende des Array-Speichers
51-52	Zeiger auf die Stringgrenze

Alle Werte werden in der LOW/HIGHbyte Darstellung abgelegt, d.h. das Ergebnis errechnet sich aus dem ersten Byte plus dem zweiten Byte mal 256. Es ist zu beachten, daß die Strings von oben beginnend nach unten abgespeichert werden, die anderen Daten von unten beginnend immer weiter hoch abgespeichert werden. Wenn sich beide treffen, ist kein Speicher mehr frei. Laden Sie also das Programm mit den niedrigeren Zeilennummern. Setzen Sie den BASIC-Start auf das Ende dieses Programmes:

```
POKE 43,PEEK(45)-2:POKE 44,PEEK(46)
```

(Falls, was sehr unwahrscheinlich ist, PEEK(45) 1 oder 0 ist, so geben Sie ein:

```
POKE 44,PEEK(46)-1)
```

Nach dem Laden des zweiten Programmes müssen Sie jetzt den BASIC-Anfang wieder heruntersetzen:

```
POKE 43,1:POKE 44,8
```

und schon haben Sie beide Programme zusammen im Speicher, und können diese jetzt als Gesamtprogramm abspeichern.

4.3 "Befehlserweiterungen"

Neben der geringen Geschwindigkeit und dem schlechten Editor ist ein weiterer Nachteil von BASIC 2.0 die Befehlsarmut. Als Gegenmittel kann man eine BASIC-Erweiterung - wie das sogar unter GEOS laufende BECKERbasic - benutzen. Viele Features lassen sich jedoch auch durch geschicktes Ausnutzen der vorhandenen Befehle "herbeizwingen", wozu Sie die folgenden Tips anregen sollen.

4.3.1 Positive Zahl bei FRE(0)

Sicher haben Sie es schon bemerkt: Die FRE(0)-Funktion funktioniert nicht ganz richtig, was daran liegt, daß das gesamte C64-Betriebssystem ursprünglich für den VC20 geschrieben wurde und ohne Änderungen auch im C64 Verwendung fand. Nun hatte der VC20 aber keine 64 KByte, sondern in der Grundausstatung nur schlaffe 5 KByte. Daher reichte es aus, für die FRE(0)-Routine eine Integer-Zahl als Ergebnis zu programmieren, die einen Wertebereich von ca. -32000 bis +32000 hat. Das erste Bit gibt das Vorzeichen an.

Wenn nun im C64 etwa 38 KByte frei sind, so wird in der Binärzahl genau dieses Bit gesetzt, was in der Bildschirmausgaberoutine die negative Zahl zur Folge hat. Um dies zu vermeiden, addieren Sie einfach 65536 zu dem Ergebnis, und Sie erhalten den wirklich freien Speicher. Beispiel:

```
10 DEF FNFR(X)=FRE(0)-65536*(FRE(0)<0)
20 PRINT FR(0)
30 A$="KKKKKKKK":A$=A$+A$
40 PRINT FR(0)
```

4.3.2 Wahrheitswerte

Mit dem merkwürdigen "<0" in der oben vorgestellten Routine hat es folgende Bewandtnis. Das BASIC des C64 macht im Gegensatz zu anderen Sprachen keinen Unterschied zwischen logischen- und arithmetischen Operationen. Sogenannte Wahrheits-

werte sind beim C64 einfach Zahlen, und zwar 0 für falsch und -1 (man beachte das Minus) für wahr. Diese Tatsache kann sich der Programmierer an diversen Stellen zunutze machen, ein Beispiel ist die oben erwähnte erweiterte FRE(0)-Funktion.

Wenn das Ergebnis der alten FRE(0)-Routine bereits größer als null ist (was dann auftritt, wenn weniger als 32 KByte Speicher frei sind, die dann in den 15 vorhandenen Bits dargestellt werden können, s.o.), dann ist der Ausdruck (fre(0)<0) gleich 0, d.h.:

```
fr(x)=fre(0)-65536*0=fre(0).
```

Wenn fre(0) negativ ist, d.h. wenn Bit 15 gesetzt ist (es hat sich eingebürgert, beim Zählen von Bits immer von rechts nach links vorzugehen, und mit null - nicht etwa mit eins - anzufangen. Bit 15 ist daher das 16. Bit), dann ist der Ausdruck (fre(0)<0) gleich -1, d.h.:

```
fr(x)=fre(0)-65536*-1=fre(0)+65536,
```

und genau dieses Ergebnis war gewünscht. Auf der anderen Seite kann man auch ganz normalen Variablen Wahrheitswerte zuweisen, um mit diesen dann entweder weiterzurechnen oder sie z.B. in einer If-Then-Abfrage zu benutzen.

```
10 a=b>10
20 b=b+1
30 print b
40 if not a then 10
```

In diesem Zusammenhang weise ich darauf hin, daß nicht nur die Vergleichsoperatoren '>' und '<', sondern alle logischen Operatoren, die auch nach einem If stehen können, als normale Rechenzeichen betrachtet werden können; natürlich auch das Gleichheitszeichen selbst. So bewirkt die Zuweisung a=b=10 bei einigen BASIC-Dialekten, daß sowohl a als auch b auf 10 gesetzt werden.

Nicht so beim Commodore 64: Hier wird bei dieser Befehlsfolge das erste Gleichheitszeichen als Zuweisung interpretiert, das zweite Gleichheitszeichen aber als logischer Vergleichsoperator, was bedeutet, daß der Wert von b überhaupt nicht geändert wird und der Wert von a dann -1 wird, wenn b vorher 10 war, ansonsten erhält A den Wert 0. Auch wenn sich dies auf den ersten Blick recht umständlich aussieht, steht dem Programmierer hiermit ein mächtiges Programmierwerkzeug zur Verfügung.

4.3.3 Bildschirmdarstellung

Auch wenn die Bildschirmdarstellung gewöhnlich nicht die Hauptsache bei einem Programm ist, ist sie doch eine wichtige Nebensache, da diese eine Hilfe ist, den Anwender Ihrer Programme "bei der Stange" zu halten. Im folgenden will ich nun einige Bildschirmeffekte und Hilfsroutinen vorstellen, von denen sich sicher einige auch in Ihren Programmen verwenden lassen.

Bildschirmlöschen

Mit folgendem Befehl kann jedes Zeichen des Bildschirmes direkt verändert, also z.B. auch gelöscht werden.

```
POKE 1024+X+Y*40,Z
```

wobei X und Y die Koordinaten sind. Es ist zu beachten, daß der Ursprung sich entgegen den normalen mathematischen Gepflogenheiten links oben befindet. Z ist das neue Zeichen im Bildschirmcode. Dieser Code ist der Tabelle im Anhang zu entnehmen. Bei älteren Modellen muß zusätzlich noch ein Farbwert gepoket werden, hierzu verweise ich aber auf das mitgelieferte Commodore-Handbuch.

Die Tatsache, daß die Y-Koordinaten oben beginnen und nach unten hin immer größer werden, ist übrigens bei Homecomputern allgemein üblich, und rührt wohl aus der Tatsache her, daß der Rasterstrahl des Fernsehers oder Monitors Zeile für Zeile von oben nach unten geht und in jeder Zeile von links nach rechts, so daß im (Bildschirm-)Speicher die einzelnen Buchsta-

ben oder Punkte auch dementsprechend angeordnet sein müssen. Außerdem entspricht diese Anordnung der normalen Schreibweise in lateinischer Schrift, also von oben nach unten und links nach rechts

Ein Beispiel hierfür wurde bereits unter Kapitel 4.2.1. geliefert, wenn man in dem Poke für den Wert 1 die Zahl 32 einsetzt, so wird der Bildschirm langsam von links nach rechts gelöscht. Diese Routine kann man stark variieren, hier nur 3 Beispiele als Anregung:

1. Beispiel

```
10 FOR X=0 TO 19
20 FOR Y=0 TO 24
30 POKE 1024+X+Y*40,32
40 POKE 1063-X+Y*40,32
50 NEXT Y,X
60 PRINT "<CLR-HOME>"
```

2. Beispiel

```
10 FOR Y=0 TO 12
20 FOR X=0 TO 39
30 POKE 1024+X+Y*40,32
40 POKE 1984+X-Y*40,32
50 NEXT X,Y
60 PRINT "<CLR-HOME>"
```

3. Beispiel

```
10 FOR K=0 TO 12: REM EVT.STEP 0.5,STEP 0.25 O.Ä.
20 FOR X=K*1.6 TO 39-K*1.6
30 POKE 1024+X+K*40,32
40 POKE 1984+X-K*40,32
50 NEXT X
60 FOR Y=K TO 24-K
70 POKE 1024+K*1.6+Y*40,32
80 POKE 1063-K*1.6+Y*40,32
90 NEXT Y,K
100 PRINT "<CLR-HOME>"
```

Bitte beachten Sie, daß Sie nach Ihren Löschroutinen immer ein Print "<CLR-Home>" setzen sollten, da der Bildschirm zwar optisch gelöscht sein mag, aber nicht unbedingt auch alle Zeiger auf ihn.

Z.B. werden durch die vorgestellten Routinen die Tabelle mit den Doppelzeilen, also die Zeilen, in denen ein Return zur Folge hat, daß man sich 2 Zeilen nach unten bewegt, weil hier z.B. eine BASICzeile mit mehr als 40 Zeichen steht, nicht gelöscht. Natürlich ist das einzelne "Auspoken" von Buchstaben recht langsam, ich stelle daher hier eine Routine vor, mit der man eine beliebige Zeile löschen kann:

```
poke 781,zeile: sys 59903
```

Beispiel:

```
10 for t=0 to 12
20 poke 781,t:sys 59903
30 poke 781,24-t:sys 59903
40 next t
```

Ausschalten des Bildschirmes

Mit einem einfachen Poke ist es möglich, den Bildschirm so auszuschalten, daß man auf dem Fernseher oder Monitor nicht mehr sieht, intern aber der Bildschirmspeicher nicht gelöscht wird. Wozu diese Spielerei gut ist, fragen Sie? Zunächst einmal werden die Zugriffe des VIC (Video-Chip) auf das RAM unterdrückt, was zur Folge hat, daß der Prozessor gleichmäßiger (und unwesentlich schneller) arbeiten kann, was z.B. beim Laden von der Datasette ausgenutzt wird. Dies ist auch dann nützlich, wenn Sie langwierige Berechnungen ausführen lassen wollen, und den Computer dafür über Nacht rechnen lassen. Dann ist wohl eine Bildschirmausgabe reichlich überflüssig.

Auch ergibt sich bei Benutzung des Pokes ein etwas besserer Sound, dies wird in Kapitel 7 näher ausgeführt. Eine wesentliche Eigenschaft dieses Pokes ist, daß hiervon nur der VIC als solcher betroffen ist und die sonstigen Vorgänge im Computer wie gewohnt weiterlaufen. Es können also durchaus bei ausgeschaltetem Bildschirm Print-Operationen oder Graphik-Befehle ausgeführt werden, die bei Wiedereinschalten des Bildschirmes schlagartig zu sehen sind, was z.B. bei Titelbildern sehr effektvoll sein kann.

```
POKE 53265,11 schaltet den Bildschirm aus,
POKE 53265,27 schaltet ihn wieder an.
```

Genauer gesagt wird der Bildschirm durch das Bit 4 der Speicherstelle 53265 ein- und ausgeschaltet. Im normalen Textmodus reichen daher die obigen Pokes. Allerdings handelt es sich bei dieser Speicherstelle um ein allgemeines Steuerregister, das auch noch für andere Aufgaben, insbesondere für den Bildschirmmodus (Text oder Grafik, Multicolor-Modus) zuständig ist. Sofern Sie auch diese Möglichkeiten nutzen wollen, müssen Sie folgende Pokes benutzen:

```
POKE 53265,PEEK(53265) AND 239  Bildschirm aus
POKE 53265,PEEK(53265) OR 16    Bildschirm an
```

Übrigens kann die Farbe des ausgeschalteten Bildschirmes über das Rahmenregister gesteuert werden:

```
POKE 53280,C
```

setzt den ausgeschalteten Bildschirm auf die Farbe C.

Textformatierungen

Hier sollen nun kleine Hilfsroutinen vorgestellt werden, mit denen auf komfortable Art und Weise Texte ausgegeben werden können:

Cursorpositionierung

Um an einer ganz bestimmten Stelle des Bildschirmes etwas auszugeben, kann man natürlich Print"<Home>,<Cursor unten>,<Cursor unten>,<Cursor unten>,<Cursor rechts>" usw. benutzen, was jedoch sehr unübersichtlich wird, und spätestens dann, wenn die Werte erst während des Programmes berechnet werden, erhält man sehr langsame, speicherintensive und konfuse Programme. Dabei stellt das Betriebssystem eine einfache Routine zur Cursorpositionierung bereit:

```
Poke 214,Zeile:Poke 211,Spalte:Sys 58732
```

Beispiel:

```
10 FOR W=0 TO 2*π STEP .1
20 POKE 214,15+14*COS(W):POKE 211,12+12*SIN(W)
30 SYS 58732
```

```
40 PRINT "DATA BECKER";
50 NEXT W
```

Umgekehrt kann es natürlich auch sinnvoll sein, feststellen zu können, wo sich der Cursor augenblicklich (z.B. nach einer Input-Anweisung) befindet. Hierbei benötigt man keinen Sys-Befehl, ein einfacher Peek der beiden Speicherstellen reicht aus:

```
PRINT "ZEILE:";PEEK(214)
PRINT "SPALTE:";PEEK(211)
```

Stringausgabe

Von modernen Textverarbeitungsprogrammen ist man gewohnt, Text rechts- oder linksbündig, zentriert oder in Blocksatz ausgeben zu lassen:

```
Dieser Text hier
ist
linksbündig

                                        Was Sie hier lesen,
                                                        ist
                                               rechtsbündig.

                         Diese Worte
                           wurden
                    zentriert ausgegeben

Was  Sie  hier  lesen,
das  ist  ein  Text  im
Blocksatz.
```

Auch wenn der normale Print-Befehl diese Möglichkeiten nicht bietet, können Sie mit Hilfe der im folgenden vorgestellten Routinen auch Ihre Texte formatiert ausgeben. Alle hier vorgestellten Routinen bedürfen eines Eingabestrings, der höchstens 40 Zeichen lang sein darf. Sofern Sie längere Texte ausgeben wollen, ist eine Word-wrap-Routine von Vorteil: Es wird hierbei angenommen, daß im String TX$ der auszugebende Text steht:

```
10000 if len(tx$)<40 then 10070
10010 t=41
10020 if mid$(tx$,t,1)=" " then 10050
10030 t=t-1: if t>=1 then 10020
10040 t$=left$(tx$,40):tx$=right$(tx$,len(tx$)-40):goto10060
10050 t$=left$(tx$,t-1):tx$=right$(tx$,len(tx$)-t)
10060 gosub 20000:REM In 20000 muß die ausgaberoutine stehen
10070 if len(tx$)<=40 then t$=tx$:goto 20000
10080 goto 10010
```

Diese Routine ist ein Unterprogramm und muß mit gosub 10000 angesprungen werden. Die Funktionsweise dieser Routine bedarf wohl mit Ausnahme der Zeile 10070 keiner weiteren Erklärung. Ein Goto auf ein Unterprogramm ? Und warum ist das eigentliche Unterprogramm nicht mit Return abgeschlossen?

Es handelt sich hierbei um einen Programmiertrick, der auch im Betriebssystem des C64 häufig benutzt wird. Der Gosub-Befehl legt die Adresse des zur Zeit abgearbeiteten Befehls auf einem Zwischenspeicher - dem sogenannten Stack - ab und springt zu der angegebenen Zeilennummer. Der Return-Befehl holt sich nun von dem Stack die Ursprungsadresse, um dort das Programm fortzusetzen. Wenn nun von einem Unterprogramm aus ein anderes angesprungen wird, um es dann zu beenden, könnte man schreiben:

```
GOSUB Zeilennummer:RETURN
```

Den gleichen Effekt hat aus oben beschriebenen Gründen die Sequenz GOTO Zeilennummer. Nun folgen die möglichen Routinen für 20000:

Linksbündig

```
20000 PRINT T$;:IF LEN(T$)<40 THEN PRINT
20010 RETURN
```

Rechtsbündig

```
20000 PRINT SPC(40-LEN(T$));T$;:RETURN
```

Zentriert

```
20000 PRINT TAB(20-LEN(T$)/2);T$;:IF LEN(T$)<40 THEN PRINT
20010 RETURN
```

Blocksatz

```
20000 I=1:SP(0)=0:FOR T=1 TO LEN(T$)
20010 IF MID$(T$,T,1)=" " THEN SP(I)=T:I=I+1
20020 NEXT T:SP(I)=LEN(T$)+1
20025 IF I=1 THEN PRINT T$:RETURN
20030 A=(I+38-LEN(T$))/(I-1):R=INT((A-INT(A))*(I-1)+.5)
20040 FOR T=0 TO I-2
20050 PRINT MID$(T$,SP(T)+1,SP(T+1)-1-SP(T));SPC(A-(R>0));
20060 R=R-1: NEXT T
20070 PRINT RIGHT$(T$,SP(I)-SP(I-1));:RETURN
```

Bei der Blocksatzroutine muß im Hauptprogramm das Feld sp() dimensioniert werden, z.B. durch die Zeile:

```
10 DIM SP(40)
```

Diese Routinen sollten nun nicht als komplette Programme zum Abtippen verstanden werden (wenngleich ich Sie daran nicht hindern kann und will), sondern als Anregungen für eigene Programme. Denkbar wäre z.B. in Zeile 20000 nicht eine der 4 Routinen zu schreiben, sondern eine Verzweigungsroutine, die in Abhängigkeit einer Modus-Variablen, z.B. MO, in die jeweilige Routine verzweigt, die dann natürlich bei anderen Zeilennummern stehen müssen. Beispiel:

```
20000 ON MO GOTO 20100,20200,20300,20400
```

Reverse Zeichen

Bleiben wir bei der Ausgabe: Der C64 hat eine sehr einfache Möglichkeit, die Bildschirmzeichen revers darzustellen, indem man nach dem Print-Befehl dem ersten Anführungszeichen die Taste RVS-On drückt und dann den Text schreibt. Die Sache hat nur einen Schönheitsfehler. Große Buchstaben schließen im 8x8-Punkte-Zeichensatz direkt mit dem oberen Rand ab, so daß bei inverser Darstellung oben eine Zeile zu fehlen scheint.

Probieren Sie es doch einfach mal aus, schalten Sie im Direktmodus den Revers-Modus mit Control-9 ein, tippen z.B. "TEST", und Sie sehen, was ich meine. Als Lösung bietet sich geradezu an, in der Zeile darüber die Unterstreichungssymbole Commodore-@ auszugeben. Beispiel:

```
10 Print "________________"
20 Print "<RVS-ON>DIES IST EIN TEST"
```

Scroll-Routine

Wer hat sie nicht schon immer gebraucht: eine Routine, um den Bildschirm nach oben schieben zu können. Überflüssig und zumal umständlich wäre es, diese direkt in BASIC zu programmieren, zudem es hierfür einen einfachen Sys-Befehl gibt, den ich Ihnen hier nun vorstellen möchte. Mit diesem Befehl wird der Bildschirm nach oben gescrollt und die unterste Zeile gelöscht:

```
sys 59626
```

4.3.4 Fakultät

Für die im Commodore-Befehlssatz nicht vorhandene Fakultätsfunktion möchte ich Ihnen hier 3 Routinen vorstellen, die diese Funktion ersetzen. Zunächst streng nach der Definition:

```
10000 F=1
10010 FOR T=2 TO N
10020 F=F*N
10030 NEXT T
10040 RETURN
```

Diese Routine berechnet für ein gegebenes n die Fakultät f. Der Nachteil ist, daß dies nur für recht kleine n funktioniert, da irgendwann die größte vom C64 darstellbare Zahl (ca. 10^38) erreicht wird. Abhilfe schafft die sog. asymptotische Darstellung:

```
log n! = log 1 + log 2 + ... + log n
```

Der Logarithmus von n! ist auch für große n noch berechenbar.

```
10000 F=0
10010 FOR T=1 TO N
10020 F=F+LOG(T)
10030 NEXT T
10035 F=F/LOG(10)
10040 PRINT 10^(F-INT(F))"E";INT(F)
10050 RETURN
```

Falls nötig, kann man den Print-Befehl durch eine entsprechende Variablenzuweisung ersetzen. Bleibt noch der Nachteil, daß die Ausführung für große n sehr langsam wird. Hierbei hilft eine mathematische Umformung, auf die ich jetzt nicht weiter eingehen möchte, sondern deren Ergebnis ich mitteile:

```
log n!≈(n+0.5)*log n - n + 1/(12*n) + log sqr(2*π)
```

```
10000 F=((N+0.5)*LOG(N)-N+1/(12*N)+LOG(SQR(2*π)))/LOG(10)
10010 PRINT 10^(F-INT(F))"E";INT(F)
10020 RETURN
```

Die letzte Formel kann nun nach zwei weiteren Umformungen auch in die Form einer deffn-Anweisung gesteckt werden:

```
DEFFN FA(N)=INT(SQR(2*π*N)*N^N*EXP(-N+1/(12*N))+.5)
```

4.3.5 Integer- und LOG-Routine

Bleiben wir bei der Mathematik: Leider halten sich die Int- und LOG-Funktion des C64-BASIC nicht an die allgemeinen Gepflogenheiten. Die Integer-Routine weicht von der normalerweise verwendeten ab, was das Verhalten im negativen Bereich angeht. So ergibt z.B. int(-π) nicht etwa, wie erwartet, das Ergebnis -3, sondern -4.

Das mag nicht weiter schlimm sein, wenn man dies weiß und nur auf dem C64 programmiert, falls man jedoch z.B. Algorithmen aus Büchern übernehmen möchte, so kann diese Funktion schon zu Verwirrungen führen. Abhilfe schafft folgende Funktions-Definition:

```
DEFFN IN(X)=INT(ABS(X))*SGN(X)
```

Die LOG-Funktion ist nicht, wie man bei der Abkürzung leicht glauben könnte, die Logarithmus-Funktion zur Basis 10, sondern zur Basis e. Der Logarithmus zur BASIS 10 ist hieraus errechenbar durch Dividieren durch LN(10), also beim C64 LOG(10), wie in den Fakultäts-Beispielen gezeigt.

4.3.6 Input ohne Fragezeichen

Hat es Sie nicht auch schon gestört? Da will man nach dem Namen fragen mit dem Befehl

```
INPUT "Bitte geben Sie hier Ihren Namen ein: ",N$
```

und der Rechner reagiert darauf mit einem

```
Bitte geben Sie hier Ihren Namen ein: ?
```

Es gibt verschiedene Methoden, dieses Fragezeichen zu unterdrücken, die einfachste und sicherste Methode scheint mir zu sein, einen sogenannten Tastaturkanal zu öffnen:

```
10 OPEN 1,0
20 PRINT "BITTE GEBEN SIE HIER IHREN NAMEN EIN: ";
30 INPUT #1,N$
40 CLOSE 1
```

Es reicht aus, den Open-Befehl nur einmal am Anfang Ihres Programmes ausführen zu lassen und darauf beliebig viele INPUT-Befehle ausführen zu lassen. Der CLOSE-Befehl muß dann vor Beendigung des Programmes stehen.

4.3.7 Get# fünfmal schneller

Wenn Sie mit Get# Daten von der Diskette einlesen, so können Sie bei längeren Files (unter länger kann man hier schon Files über 2 KBytes verstehen) erst einmal in Ruhe Kaffee trinken

gehen; der Befehl ist geradezu unverschämt langsam. Eine wesentliche Beschleunigung bietet folgender Tip:

- Öffnen Sie das File wie gewohnt.
- Poken Sie die Filennummer nach 781 (X-Register).
- Rufen Sie die Eingabe-Umleitungsroutine (CHKIN) auf durch Sys 65478.
- Lesen Sie Ihr File mit dem normalen Get-Befehl ein.
- Setzen Sie den Get-Befehl mit Sys 65484 auf die Tastatur zurück und schließen das File.

```
10 OPEN 1,8,5,"FILE,S,R"
20 POKE 781,1:SYS 65478
30 FOR T=1 TO 2048
40 GET A$:PRINT A$;
50 NEXT T
60 SYS 65484:CLOSE 1
```

4.3.8 Cursorblinken bei GET

Gehen wir zurück zu dem "normalen" Get-Befehl von der Tastatur. Oft ist es nötig, mit diesem einen Input-Befehl zu simulieren, z.B. um besondere Zeichen abzufragen (etwa das Komma). In diesem Fall ist es recht lästig, daß der Cursor nicht zu sehen ist, und der Anwender gar nicht weiß, daß er etwas eingeben kann, schließlich ist er ja gewohnt, daß der Cursor bei Eingaben blinkt. Mit einem einfachen Poke kann jederzeit, also auch bei beliebigen anderen Befehlen, der Cursor zum Blinken gebracht werden:

```
POKE 204,0:     Cursor ein
POKE 204,1:     Cursor aus.
```

Das folgende Beispiel simuliert eine Input-Anweisung, wobei auch die Trennzeichen (, und :), die normalerweise einen "Redo from start"- oder "extra ignored"-Error hervorrufen und den String nicht komplett einlesen, mit eingelesen werden:

```
10 IN$="":POKE 204,0
20 POKE 198,0:WAIT 198,1
30 GET A$: PRINT A$;:IF A$=CHR$(13) THEN 50
40 IN$=IN$+A$:GOTO 20
50 POKE 204,1
60 PRINT "EINGABE: "IN$
```

Dieses Beispiel verdeutlicht natürlich nur das Prinzip, Sondertasten wie INST/DEL o.ä. funktionieren nicht.

4.3.9 Abfrage von Sondertasten

Sicher wollten Sie schon einmal die Sondertasten Shift/Control/C= von Ihrem Programm aus abfragen, aber haben gemerkt, daß dies mit normalen BASIC-Befehlen nicht möglich ist, da bei GET oder INPUT erst dann ein Wert zurückgeliefert wird, wenn diese Tasten zusammen mit anderen Tasten gedrückt werden. Wenn man weiß, wo das Betriebssystem diese Tasten ablegt, ist nur noch ein einfacher Peek nötig. In der Speicherstelle 653 steht folgender Code:

```
1=Shift, 2=Commodore, 4=Ctrl.
```

Wenn mehrere dieser Tasten gleichzeitig gedrückt werden, addieren sich die Werte.

```
10 a=peek(653)
20 print "<Home>";
30 if a and 1 then print "Shift ";:goto 50
40 print "      ";: rem 6 Leerzeichen
50 if a and 2 then print "Commodore ";:goto 70
60 print "          ";: rem 10 Leerzeichen
70 if a and 4 then print "Control";:goto 10
80 print "       ";:rem 7 Leerzeichen
90 goto 10
```

Mit dieser Methode können Sie nun bis zu 16 Funktionstasten abfragen, indem Sie mittels eines normalen GET-Befehles die Funktionstaste einlesen und sofort danach überprüfen, ob gleichzeitig die Commodore- oder Control-Taste gedrückt wurde.

4.3.10 Abfrage von mehreren Tasten gleichzeitig

Hatten Sie auch schon einmal folgendes Problem? Da schreibt man ein Programm (z.B. ein Spiel), fragt regelmäßig die Tastatur ab, wobei z.B. mit den Cursortasten die Bewegung des Spielers angegeben wird, gleichzeitig kann er sich z.B. mit der Space-Taste unsichtbar machen. Mit normalem BASIC ist das in dieser Form nicht möglich, da bei GET immer nur eine Taste abgefragt wird, man also erst einmal die Cursor-Taste loslassen müßte, bevor man sich unsichtbar machen könnte; und das, wo der Verfolger gerade so dicht hinter einem ...

Wie Sie sich nach dieser Einleitung eigentlich schon denken können, ist es über die entsprechenden Peeks auch möglich, beliebige Tastenkombinationen abzufragen. Leider ist das aber nicht so einfach wie z.B. bei der Shift/Control/C=-Tasten. Deshalb muß ich etwas weiter ausholen. Die Tastatur des C64 besteht aus 66 Tasten, wobei 2 Tasten aus der Reihe fallen, nämlich die Restore- und Shift-Lock-Taste. Die Restore-Taste ist direkt mit der entprechenden Leitung des Rechners verbunden und löst einen Interrupt aus (einen sog. NMI, nicht-maskierbarer-Interrupt, d.h. man kann ihn softwaremäßig nicht ausschalten).

Die Shift-Lock-Taste ist mit der linken Shift-Taste parallel geschaltet, so daß der Rechner nicht erkennen kann, ob jetzt die Shift-Lock-Taste eingerastet oder die linke Shift-Taste gedrückt ist. Warum ich das Wort "linke" so betone? Es ist durchaus möglich, mittels eines Peeks zu unterscheiden, ob die linke oder rechte Shift-Taste gedrückt wurde. Die restlichen 64 Tasten sind in einer 8x8-Matrix wie in der Abbildung vernetzt:

Die jeweiligen "Enden" sind nun mit den Registern 56320 und 56321 der CIA 1 verbunden (wobei in Mittelamerika niemand vor diesen Chips Angst zu haben braucht, die Abkürzung steht schlicht und einfach für Complex-Interface-Adapter; von diesen Chips gibt es im C64 2 Stück). Das Register 56320, auch Port A genannt, ist im C64 als Ausgang und Port B (56321) als Eingang geschaltet.

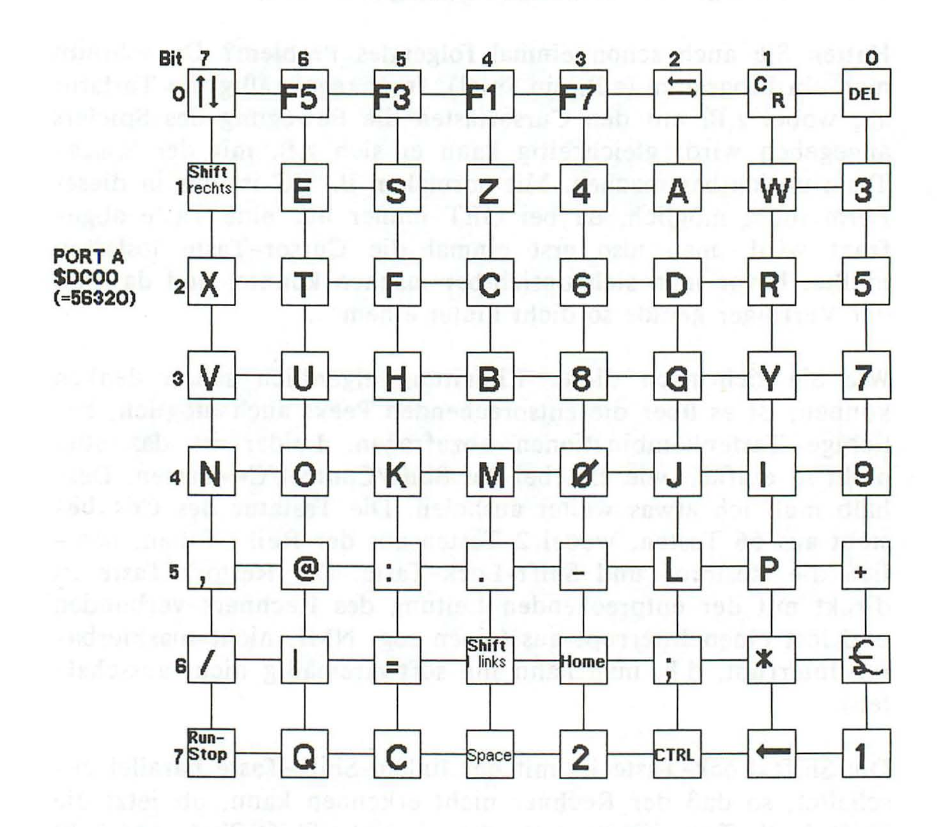


Abb. 3 Tastaturmatrix

Werden nun alle Pins des Ports A auf Low gesetzt und ist eine Taste gedrückt, so pflanzt sich das Low-Signal fort, und das entsprechende aus der Tabelle zu entnehmende Bit in Port B wird Low, die anderen High. Dies ist dadurch zu erklären, daß in der CIA ein physikalisches gesetztes Bit logisch als Low angesehen wird. Vielleicht war Ihnen das jetzt zu trocken, daher erkläre ich das Ganze hier noch mal an einem Beispiel.

Um die Tastatur abzufragen, wird zunächst Port A auf 0 gesetzt (Poke 56320,0). Nehmen wir an, jetzt wurde die Taste C gedrückt. Das heißt, die logische 0 von Port A wird über die gedrückte Taste auf Bit 4 von Port B übertragen. Dort erhält man durch Peek(56321) das Ergebnis 239. Diese 239 kam dadurch zustande, daß alle Bits außer dem vierten Bit gesetzt sind, also 255-2^4=255-16=239.

Nun wissen Sie also, daß die gedrückte Taste eine der Tasten F1, Z, C, B, M, Punkt, rechte Shifttaste, oder Leertaste war, denn alle diese Tasten hätten das 4. Bit auf Low gesetzt. Um herauszufinden, welche von den acht Tasten der Übeltäter war, probieren Sie einfach alle der Reihe nach durch, d.h. Sie setzen in Port A nacheinander jeweils alle Bits auf eins, außer dem gerade zu testenden. Zweckmäßigerweise fängt man bei Bit 0 an: Poke 56320,255-2^1. Jetzt ist Peek(56321) nicht mehr 239, sondern 255, d.h. daß es nicht die F1-Taste war, die gedrückt wurde. Weiter:

```
Poke 56320,255-2^2
```

Auch hier ist

```
Peek(56321)=255.
Poke 56320,255-2^3
```

Jetzt haben wir Glück: Peek(56321) wird 239, und wir können die Suche abbrechen, wir wissen jetzt, daß die C-Taste gedrückt wurde. Langer Rede, kurzer Sinn: Natürlich sollen diese Ausführungen nicht zeigen, wie man feststellt, ob man die Taste C gedrückt hat, daß ginge mit dem Befehl

```
GET A$:IFA$="C" THEN PRINT "C GEDRÜCKT"
```

schneller und einfacher, sondern zeigen, wie man mehrere Tasten gleichzeitig abfragt. Angenommen, Space und Cursor rechts wurden gleichzeitig gedrückt. Nach dem ersten Nullsetzen des Port A erhalten wir als Port B 235=255-2^4-2^2. Für Port A=255-2^0=254 ergibt sich für Port B=251=255-2^2, für Port

A=255-2^7=127 ergibt sich für Port B=239=255-2^4. Für alle anderen Pokes in A ist Port B 255. Hieraus ergibt sich, daß die Tasten (0|2) und (7|4) gedrückt wurden.

Sie sehen schon, daß es eigentlich überflüssig ist, zunächst den Port A auf null zu setzen, da dies für die eigentliche Tastenabfrage nicht mehr von Belang ist, es ist aber dann von Vorteil, wenn keine Taste gedrückt wurde, Port B also 255 ist, da man dann sofort aufhören und sich die zeitintensiven weiteren Verzweigungen sparen kann.

Noch ein praktischer Hinweis am Schluß: Wenn Sie von BASIC aus an die Sache rangehen, werden Sie, nur mit diesen Hinweisen bewappnet, eine ziemliche Bauchlandung erleben, da sich das Betriebssystem für seine Tastaturabfrage natürlich auch der beschriebenen Register bedient, und diese wird sich dauernd verändern, denn das Betriebssystem fragt die Tastatur etwa 60mal in der Sekunde ab. Mit dem Poke

```
POKE 56334,0
```

läßt sich der Interrupt allerdings ausschalten, so daß Sie "freie Bahn" haben, vergessen Sie aber nicht, diesen nach der Abfrage mit

```
POKE 56334,1
```

vor Beendigung des Programmes wieder anzuschalten, da Sie ohne Tastaturabfrage wohl kaum noch etwas mit Ihrem C64 anfangen können. Zum Schluß noch einmal für alle, die trotz (oder wegen ?) meiner ausführlichen Erklärungsversuche immer noch nichts von diesem wirklich nicht einfachen Thema verstanden haben, ein kurzes Beispielprogramm.

```
10 POKE 56334,0:PRINT CHR$(147)
20 FOR A=0 TO 7
30 : POKE 56320,255-2^A
40 : B=PEEK(56321)
50 : IF B=255 THEN 90
60 : FOR T=0 TO 7
70 : : IF (B AND 2^T)=0 THEN PRINT"(";A;",";T")",
```

```
80 : NEXT T
90 NEXT A
100 POKE 56334,1
110 GOTO 10
```

Dieses Programm gibt die Rasterkoordinaten der gedrückten Tasten aus, wie sie aus der Tabelle zu entnehmen sind. Hierbei sehen Sie auch gleich die Grenzen dieser Abfragemethode, bei zwei Tasten gleichzeitig beträgt die Trefferquote noch 100%, bei drei Tasten gleichzeitig gibt es schon Fälle, bei denen eine Taste zuviel gemeldet wird (z.B. Gedrückt: 1,2,3. Gemeldet: 1,2,3,4).

4.3.11 Momentan gedrückte Taste

Und gleich weiter mit der Tastaturabfrage. Oft ist es nötig zu wissen, welche Taste gerade in diesem Augenblick gedrückt ist. Mit dem GET-Befehl erhalten Sie nämlich auch solche Tasten geliefert, die vor einer halben Stunde gedrückt wurden, wenn zwischendurch die Tastatur nicht abgefragt wurde. Wenn Sie nur diese Langzeitwirkung ausschalten wollen, so reicht der Poke

```
Poke 198,0
```

vor Ihrem GET-Befehl. Mit diesem Poke wird nämlich der Tastaturpuffer gelöscht, der C64 "vergißt" alle bisher gedrückten Tasten. Wenn eine Taste längere Zeit gedrückt bleibt und diese schon einmal mit GET abgefragt wurde, bekommt man sie bei dem nächsten GET nicht noch einmal geliefert. (siehe auch nächstes Kapitel). So kann es sein, daß im Augenblick der Tastaturabfrage zwar eine Taste gedrückt ist, man dies aber mit dem GET-Befehl nicht erfahren kann.

Man könnte jetzt natürlich auf die im letzten Kapitel beschriebenen Tricks zurückgreifen, dies wäre aber recht umständlich. Wie schon gesagt wurde, fragt auch das Betriebssystem regelmäßig die Tastatur ab. Die Tastenwerte speichert es, und löscht den Speicher auch nicht, nachdem es mit der Abfrage fertig ist, so daß wir dort nachsehen können, welche Taste gedrückt ist. Mit

```
taste=PEEK(203)
```

oder

```
taste=PEEK(197)
```

erfährt man die derzeit gedrückte Taste, allerdings leider nicht den ASCII-Code derselben, sondern den sogenannten Scan-Code. Dieser ist naturgemäß eng mit der Tastaturmatrix verbunden, der Scan-Code einer beliebigen Taste ist

```
Scan-Code = Bit in Port A * 8 + Bit in Port B
```

Hier noch einmal die Scan-Codes alphabetisch zusammengefaßt:

A 10	K 37	U 30	SPACE 60	3 8	@ 46
B 28	L 42	V 31	* 49	4 11	PFUND 48
C 20	M 36	W 9	+ 40	5 16	^ 54
D 18	N 39	X 23	, 47	6 19	DEL 0
E 14	O 38	Y 25	- 43	7 24	RETURN 1
F 21	P 41	Z 12	. 44	8 27	CRSLR 2
G 26	Q 62	F1 4	/ 55	9 32	CRSOU 7
H 29	R 17	F3 5	0 35	: 45	CLR/HOME 51
I 33	S 13	F5 6	1 56	; 50	← 57
J 34	T 22	F7 3	2 59	= 53	RUN/STOP 63

Abb. 4 Scan-Codes

4.3.12 Taste abwarten mit WAIT

In Zusammenhang mit dem REM-Killer bin ich schon auf die Funktion des Tastaturpuffers eingegangen. Hier will ich nun eine Möglichkeit vorstellen, diesen im Programm praktisch anzuwenden. Bei dem Befehl

```
POKE 198,0:WAIT 198,1
```

wartet der Interpreter solange, bis eine Taste gedrückt wird. Dies hat den Vorteil, daß keine IF-Abfrage wie bei

```
10 get a$:if a$="" then 10
```

erforderlich ist, der Befehl also mitten in eine Zeile zwischen andere Befehle paßt. Noch ein Wort zu dem Wait-Befehl. Dieser Befehl wird gerade von Anfängern leicht mißverstanden, daher hier noch einmal zur Klarstellung. Der Wait Befehl

```
WAIT 198,N
```

wartet NICHT, bis der Wert N in der Speicherstelle 198 steht, auch wenn dies bei der 1 zufällig zusammenfällt. Wenn der Interpreter auf den Befehl WAIT 198,n trifft, so verknüpft dieser vielmehr den Inhalt der Speicherstelle 198 logisch und mit n und bricht ab, falls das Ergebnis ungleich 0 ist, andernfalls wird nochmal verglichen.

Wenn der Befehl zwei Parameter hat (z.B. Wait 198,a,b), so wird der Inhalt der Speicherstelle 198 zunächst logisch exclusiv-oder mit b verknüpft werden (d.h. daß alle Bits, die in b gesetzt sind, im Testwert vertauscht werden, aus 0 wird 1 und 1 wird 0, und alle Bits, die in b nicht gesetzt sind, im Testwert gleichbleiben). Das Ergebnis wird nun logisch und mit a verknüpft, und wenn das Gesamtergebnis ungleich 0 ist, wird abgebrochen, sonst weitergemacht. Zur Verdeutlichung hier ein BASIC-Programm, welches denselben Effekt hat wie wait ad,b,c

```
10 W=PEEK(AD)
20 IF ((W OR B) AND NOT (W AND B)) AND A THEN 40
30 GOTO 20
40 REM WEITER

(Nach der Formel a XOR b = (a or b) and not(a and b))
```

4.3.13 GOTO X

Sicher kennen Sie das: Es soll zu einer Zeile gesprungen werden, Sie wissen aber beim Schreiben des Programmes noch nicht, welche Zeile dies sein wird, denn diese wird erst während der Programmausführung berechnet. Nun können Sie zwar bei dem GOTO-Befehl beliebige Zeilennummern angeben, nicht aber Variablen oder Formeln. GOTO 20 ist durchaus möglich, goto 10+10 oder gar goto a*10 wird Ihnen Ihr C64 im günstigsten Falle mit einem Syntax-Error quittieren, ansonsten wird er irgendwo in der Speicherlandschaft rumspringen.

Natürlich wurde das alles nicht ohne Grund von Commodore so programmiert, der Grund liegt wohl in einer Richtung der theoretischen Informatik, die den GOTO-Befehl als "unstrukturiert" abtut, weil dieser zum sogenannten "Spaghetti-Code" verleitet. Sicher ist da etwas Wahres dran, und Sie sollten mit dem GOTO x sorgsam umgehen (am besten mit REM anhängen, wohin gesprungen werden kann), aber es muß immer noch die freie Entscheidung des Programmierers bleiben, in begründeten Einzelfällen auch auf solche Befehle zurückzugreifen. Im folgenden will ich Ihnen eine kleine Hilfsroutine vorstellen, die ein GOTO x und GOSUB x erlaubt:

```
10 AD=832
20 FOR T=AD TO AD+46
30 READ A:POKE T,A
40 NEXT T
50 DATA 32,253,174, 32,138,173 :rem hole komma,hole wert
60 DATA 32,247,183, 76,163,168 :rem wert nach 14,jmp goto
70 DATA 32,253,174, 32,138,173 :rem hole komma,hole wert
80 DATA 32,247,183             :rem wert nach 14
90 DATA 169,3,32,251,163       :rem platz auf stack ?
100 DATA 165,123,72,165,122,72 :rem prg-zähler auf Stack
110 DATA 165,58,72,165,57,72   :rem zeilennummer auf Stack
120 DATA 169,141,72            :rem Gosub-Code auf Stack
130 DATA 32,163,168,76,174,167 :rem jmp goto
```

Diese Routine sollte immer am Programmanfang stehen, nachdem die Routine durchlaufen ist, stehen zwei neue Befehle zur Verfügung:

```
SYS AD,x
```

entspricht GOTO X, wobei x ein beliebiger arithmetscher Ausdruck oder eine Variable sein kann

```
SYS AD+12,x
```

enspricht GOSUB X, sonst wie oben. Da keine absoluten Sprünge innerhalb des Programmes benutzt wurden, kann AD eine beliebige Zahl sein. Wichtig ist nur, daß sich ab AD 46 Bytes freier Speicher befindet. Falls der Kassettenpuffer schon anders benutzt ist, können Sie auch den freien Bereich zwischen 704 und 766 benutzen. Da der Kassettenpuffer größer ist, wird dieser von fast allen Maschinenprogrämmchen benutzt, die man in Zeitschriften o.ä. findet, in diesem Fall ändern Sie einfach die Zeile 10 in:

```
10 AD=704
```

Allerdings kann es sein, daß der Bereich von einem Sprite benutzt wird, in diesem Falle können Sie noch auf den $Cxxx-Bereich ausweichen, wählen Sie also AD irgendwo zwischen 49152 und 53200.

4.3.14 Finden von DATA-Zeilen

Da erhält man einen ?TYPE-MISMATCH-ERROR IN 90, listet die Zeile auf, und es ist absolut kein Fehler zu entdecken, lediglich ein unschuldiges

```
90 READ X
```

steht in der Zeile. Ziemlich schnell merken Sie, was passiert ist: Irgendwo in Ihren DATA-Zeilen haben Sie sich vertippt, irgendwo steht ein Wert, den der C64 nicht als Zahl identifizieren kann. Aber wo? Wenn man weiß, wo das Betriebssystem die aktuelle DATA-Zeile zwischenspeichert, kein Problem. Durch Eingabe des Befehls

```
PRINT PEEK(63)+256*PEEK(64)
```

erhalten Sie die Zeilennummer des Übeltäters auf dem Bildschirm angezeigt.

4.3.15 Text-Hardcopy

Eine schnelle Methode, den Inhalt des Bildschirmes im Textmodus auf den Drucker zu bekommen, will ich Ihnen hier vorstellen:

```
10 OPEN 1,3:OPEN 2,4
20 PRINT CHR$(19)
30 FOR Y=0 TO 24
40 FOR X=0 TO 39
50 GET #1,A$
60 PRINT#2,A$
70 NEXT X
80 PRINT#2
90 NEXT Y
100 CLOSE 1,2
```

Das Prinzip ist schnell erklärt: Der sogenannte "Bildschirmkanal" wird geöffnet. Auf dem C64 kann grundsätzlich jedes Peripheriegerät über den OPEN-Befehl angesprochen werden. Wir haben das schon bei der Tastatur beim Unterdrücken des Fragezeichens gesehen. Zu diesem Zweck hat jedes "Gerät" eine Adresse, die im OPEN-Befehl an zweiter Stelle genannt werden muß. Die erste Stelle ist die sogenannte logische Filenummer, über die man das geöffnete Gerät anspricht. Hier zunächst eine Übersicht über die Geräte und den dazugehörigen Nummern:

Geräteadresse	Gerät	Eingabe	Ausgabe
0	Tastatur	X	-
1	Datasette	X	X
2	RS-232	X	X
3	Bildschirm	X	X
4	Drucker	-	X
5	evt. 2.Drucker	-	X
6	Plotter	-	X
7	normalerweise nicht benutzt		
8 ff	Floppies	X	X

Hierbei muß gesagt werden, daß es vom C64 aus durchaus möglich ist, auf den Geräteadressen 4 und höher Eingaben zuzulassen, da es sich um denselben Port handelt, über den auch die Floppy betrieben wird. In der Praxis ist es nur nicht sinnvoll, vom Drucker etwas auslesen zu wollen. Die einzige Operation dagegen, die wirklich unmöglich ist, ist aus naheliegenden Gründen eine Ausgabe auf die Tastatur.

Weiter zum OPEN-Befehl. Neben der logischen Filenummer und der Geräteadresse, die immer angegeben werden müssen, gibt es eine Sekundäradresse, die nicht unbedingt angegeben werden muß. Einige Geräte - insbesondere die am IEC-Bus - werden hierüber gesteuert. Genaueres gibt dann aber die Gebrauchsanweisung zu den angeschlossenen Geräten her, da es dem C64 völlig egal ist, ob und - wenn ja - welche Sekundäradresse angegeben wird. Er gibt sie ungesehen und unzensiert weiter, und der Gerätehersteller kann selber entscheiden, was er mit der Sekundäradresse machen will. Nachdem nun ein Kanal geöffnet wurde, kann hierauf mit den Befehlen PRINT,GET,INPUT und CMD zugegriffen werden:

```
PRINT #LF,a$
```

gibt den Inhalt von a$ an den Kanal LF aus.

```
CMD LF
```

leitet alle folgenden Ausgaben auf den Kanal LF um. Beispiel:

```
OPEN 1,4:CMD 1:LIST
```

gibt ein Druckerlisting aus.

```
GET #LF,a$
```

liest genau 1 Zeichen aus dem Kanal LF.

```
INPUT #LF,a$
```

liest a$ aus dem Kanal LF. Es werden solange Zeichen eingelesen, bis ein CHR$(13), ein Komma oder ein Doppelpunkt folgt.

4.3.16 Unterdrückem der "READY."-Meldung

Wollen Sie nur ein Programm beenden, ohne daß ein Ready folgen soll (z.B., weil Sie noch etwas auf dem Bildschirm stehen haben und nicht wollen, daß dieses wegscrollt) so reicht ein einfacher Befehl

```
SYS 42107
```

anstatt des END-Befehls. Auf diese Weise wird der sog. BASIC-Warmstart direkt aufgerufen, während die Programmende-Routine des Betriebssystems vor dem Warmstart noch READY. ausgibt.

Wenn sich das Betriebssystem im RAM befindet, etwa, weil Sie eines der Hilfsrogramme aus Kapitel 2 eingegeben haben, können Sie die READY-Meldung auch Ihren Bedürfnissen anpassen:

```
10 A$="FERTIG"+CHR$(13)
20 FOR T=1 TO 7
30 POKE 41847+T,ASC(MID$(A$,T,1))
40 NEXT T
```

Natürlich darf a$ zusammen mit dem chr$(13) nicht länger als 7 Zeichen werden. Mit derselben Methode können auch andere Meldungen umgeändert werden. Ich nenne hier nur die Adressen nach dem Poke, die Zahl im FOR-Befehl ändern Sie bitte anhand der Stellenzahl ab:

OK:	41828	3 Stellen
ERROR:	41832	bitte beachten Sie, daß diese Error-String-konstante 7 Zeichen lang ist, da sie mit 2 Leerzeichen beginnt.
IN:	41840	auch IN beginnt mit einem Leerzeichen, 3 Stellen lang
READY:	41847	7 Stellen
BREAK:	41858	5 Stellen

4.3.17 Ändern der Fehlermeldungen

Im folgenden nun ein Programm, mit dem die Fehlermeldungen des BASIC-Interpreters abgeändert werden können, indem Sie andere DATA-Zeilen eingeben:

```
10 L=0: FOR T=1 TO 29: READ A$:L=L+LEN(A$): NEXT T
20 IF L>393 THEN PRINT L-393;"ZEICHEN ZUVIEL":END
30 RESTORE
40 AD=41374:FOR T=1 TO 29
50 READ A$:FOR I=1 TO LEN(A$)-1
60 POKE AD+I-1,ASC(MID$(A$,I,1))
70 NEXT I:POKE AD+LEN(A$)-1,ASC(MID$(A$,I,1)) OR 128
80 POKE 41766+T*2,(AD-32768)AND 255:POKE 41767+T*2,AD/256
90 AD=AD+LEN(A$)
100 NEXT T
110 DATA "TOO MANY FILES"
120 DATA "FILE OPEN"
130 DATA "FILE NOT OPEN"
140 DATA "FILE NOT FOUND"
150 DATA "DEVICE NOT PRESENT"
160 DATA "NOT INPUT FILE"
170 DATA "NOT OUTPUT FILE"*
180 DATA "MISSING FILENAME"
190 DATA "ILLEGAL DEVICE NUMBER"
200 DATA "NEXT WITHOUT FOR"
210 DATA "SYNTAX"
220 DATA "RETURN WITHOUT GOSUB"
230 DATA "OUT OF DATA"
240 DATA "ILLEGAL QUANTITY"
250 DATA "OVERFLOW"
260 DATA "OUT OF MEMORY"
270 DATA "UNDEF'D STATEMENT"
280 DATA "BAD SUBSCRIPT"
290 DATA "REDIM'D ARRAY"
300 DATA "DIVISION BY ZERO"
310 DATA "ILLEGAL DIRECT"
320 DATA "TYPE MISMATCH"
330 DATA "STRING TOO LONG"
340 DATA "FILE DATA"
350 DATA "FORMULA TOO COMPLEX"
360 DATA "CAN'T CONTINUE"
370 DATA "UNDEF'D FUNCTION"
380 DATA "VERIFY"
390 DATA "LOAD"
```

Die neuen Meldungen können durchaus auch länger sein als die alten, jedoch darf die Gesamtzahl der Buchstaben nicht erhöht werden. Wenn eine Meldung länger wird, muß dafür eine andere Meldung kürzer werden.

4.3.18 Erzwingen von Fehlermeldungen

Diese Fehlermeldungen (und natürlich auch die nicht abgeänderten) können nicht nur durch Ausführen eines falschen Befehls (was z.B. bei Verify-Error nicht einfach sein dürfte), sondern auch mit einem SYS herbeigeführt werden:

```
POKE 781,x
SYS 42039
```

Wäre das nicht ein "eleganter" Programmausstieg, etwa wenn Ihr Anwender das Paßwort nicht kennt? x ist übrigens direkt aus den Zeilennummern der BASIC-Zeilen des oben abgedruckten Programmes durch folgende Zuweisung zu errechnen.

```
x=zeilennummer/10-10
```

4.3.19 End-Reset

Durch folgenden Befehl wird der End-Vektor so umgebogen, daß nach Beenden eines Programmes ein Kaltstart ausgeführt wird:

```
POKE 768,143
```

4.3.20 Zeitverzögerungsschleifen

Ein immer wieder auftretendes Problem: Ein Programm muß für eine gewisse Zeit angehalten zu werden, z.B. um dem Benutzer Zeit zu geben, einen Text zu lesen o.ä. Normalerweise bedient man sich in einem solchem Falle einer FOR-NEXT Schleife, wie:

```
10 FOR T=1 TO 1000:NEXT T
```

Dies hat den Nachteil, daß man nie genau weiß, wie lange der Computer an dieser Schleife sitzen wird, man also um mehrfaches Ausprobieren und Korrigieren nicht umhinkommt. Viel komfortabler ist es hingegen, sich der eingebauten Uhr zu bedienen:

```
10 T=TI
20 IF TI-T<500 THEN 20
```

Man kann dabei die Zeit, die die Schleife anhält, in 1/60 Sekundenschritten einstellen, in diesem Fall 500/60=8.3 Sekunden.

4.3.21 Abfrage des Druckers

Zur Vermeidung eines DEVICE NOT PRESENT ERRORs mit der Folge des Programmabbruchs ist es sinnvoll, von BASIC aus abfragen zu können, ob der Drucker angeschlossen und Online ist. Dies geschieht einfach durch Öffnen des entsprechenden Kanals und Abfragen der Statusvariablen ST:

```
10 OPEN 1,4:CLOSE 1
20 IF ST=128 GOTO 10
30 PRINT "JETZT IST ER ONLINE"
```

4.3.22 Prioritätenliste

Kennen Sie folgendes Problem? Ist nun -2^2 für den C64 4 oder Minus 4? Gilt das Minus nur für die erste 2 oder für den ganzen Ausdruck? Durch Ausprobieren merken Sie, daß es für den ganzen Ausdruck gilt. Wie ist das dann aber mit -2+2, müßte da das Minus nicht auch für den ganzen Ausdruck gelten? Warum ist denn dann das Ergebnis 0? Nun, wie alles im Computer, ist auch ganz eindeutig geregelt.

Jede Rechenart hat einen Prioritätscode, und wenn zwei Rechenarten verschachtelt sind, ohne daß Klammern gesetzt wurden, so wird die Rechenart mit der höchsten Hierarchie zuerst ausgeführt. Diese Prioritätenliste will ich Ihnen hier vorstellen:

OR	$46 (70)
AND	$50 (80)
NOT	$5A (90)
Vergleich	$64 (100)
Addition	$79 (121)
Subtraktion	$79 (121)
Multiplikation	$7b (123)
Division	$7b (123)
Vorzeichenwechsel	$7d (125)
Potenzierung	$7f (127)

Ich möchte diese Liste noch einmal an einem Beispiel erläutern: Sie haben den Ausdruck a*b-c^d and 2-3. Der Computer rechnet nun in folgenden Schritten: zuerst c^d, weil die Potenzierung den höchsten Code hat:

```
a*b- (c^d) and 2-3
```

dann a*b, weil die Multiplikation den zweithöchsten Code hat:

```
(a*b)-(c^d) and 2-3
```

nun kommt die Subtraktion dran:

```
((a*b)-(c^d)) and (2-3)
```

und fertig sind wir. Bei Kenntnis dieser Regeln kann man eine Menge an Klammern einsparen.

4.3.23 Springen aus FOR-NEXT-Schleifen

Im BASIC V2 des C64 wird glücklicherweise der GOTO-Befehl sehr liberal gehandhabt, d.h. man kann von jeder Stelle aus überall hin springen. Daß dies nicht selbstverständlich ist, weiß jeder, der z.B. einmal in sog. strukturierten Sprachen (wie Pas-

cal) programmiert hat. Falls es dort überhaupt ein GOTO gibt, so nur innerhalb des aktuellen Blocks.

Durch die höhere Freiheit hat der Programmierer jedoch auch eine höhere Verantwortung hinsichtlich der Tatsache, daß Schleifen und Unterprogramme sauber verlassen werden. So findet man immer wieder (und das nicht nur bei Anfängern) folgende, völlig falsche Konstruktion:

```
10 FOR T=1 TO 1000000
20 GET A$:IF A$<>"" THEN 40
30 NEXT T
40 PRINT "NACH";T;"DURCHLÄUFEN WURDE EINE TASTE GEDRÜCKT"
```

Das Vertrackte daran ist, daß diese Konstruktion zunächst sogar funktioniert. Erst nachdem diese Befehlsfolge einige Male durchlaufen wurde (und das kann in Einzelfällen durchaus bis zu 100mal dauern) meldet sich der Interpreter mit einem MEMORY OVERFLOW ERROR. Der Grund liegt darin, daß bei einem FOR-Befehl die derzeitige Adresse auf den Stack gelegt wird und bei dem NEXT-Befehl die Adresse, falls das Ende noch nicht erreicht ist, wieder ausgelesen wird und dorthin verzweigt wird, anderenfalls wird der Stack-Eintrag gelöscht.

Wird nun die Schleife durch ein GOTO verlassen, wird der Stack-Eintrag nicht gelöscht, so daß sich im Laufe der Zeit eine Menge Müll auf dem Stack ansammelt, bis dieser (mit 256 Bytes recht mager bemessen) plötzlich voll ist. Als Alternative bleibt Ihnen in solchen Fällen nichts anderes übrig, als entweder auf die FOR-NEXT-Schleife zu verzichten (langsamer), oder statt des GOTO 40 ein

```
T=100000000:GOTO 30
```

zu benutzen, so daß durch das NEXT der Stack wieder gereinigt wird. Vorher muß selbstverständlich der Wert von T in einer anderen Variablen zwischengespeichert werden. Ebenso wie in einer FOR-NEXT-Schleife tritt das Problem bei Unterprogrammen auf:

```
10 REM HAUPTMENÜ
20 INPUT "WAHL";W
30 IF WAHL=1 THEN GOSUB 100
40 ...
...
100 REM 1.UNTERMENUE
110 INPUT "WAHL";W
120 IF W=0 THEN 10:REM HAUPTMENÜ
130 IF W=1 THEN 1000
140 ...
...
1000 REM 2.UNTERMENUE
1010 INPUT "WAHL";W
1020 IF W=0 THEN 10:REM HAUPTMENÜ
1030 IF W=1 THEN ...
...
```

Man ist leicht versucht, bei verschachtelten Menüverteilern einen Unterpunkt "ZURÜCK ZUM HAUPTMENÜ" zu implementieren und diesen durch ein einfaches GOTO auf das Hauptmenü zu realisieren. Dies geht recht lange gut, aber nach einigen Aufrufen kommt der bekannte MEMORY OVERFLOW. In der ersten Ebene (Zeile 120) ist dies übrigens ganz leicht durch

```
IF W=0 THEN RETURN
```

zu realisieren. In tieferen Ebenen muß ebenfalls mit RETURN gearbeitet werden, damit der Stack gereinigt wird, wobei in der nächstniedrigeren Ebene geprüft werden muß, ob dieses RETURN durch die Wahl "Hauptmenü" zustande kam, um auch hier ein RETURN auszulösen. Am besten, man benutzt eine Variable HA o.ä., die dann den Wert 1 hat, wenn zum Hauptmenü verzweigt werden soll, andernfalls den Wert 0.

Die beiden oberen Beispiele sind zwar der Praxis entnommen, haben aber den Nachteil, daß man den Stacküberlauf nicht unmittelbar sehen kann, zu oft müßten die Routinen ausgeführt werden. Daher hier ein Beispiel, das zwar nicht sehr praxisnah ist, aber daraufhin programmiert wurde, möglichst schnell einen

Stackoverflow herbeizuführen, um Ihnen zu demonstrieren, welche Gefahr in Programmen wie den beiden oberen schlummert:

```
10 GOSUB 20
20 T=T+1:IF T=1000 THEN END
30 GOTO 10
```

Auch diese auf den ersten Blick einwandfreie Konstruktion führt nicht zu einer Zeitverzögerungsschleife, sondern zu dem bereits bekannten MEMORY OVERFLOW ERROR.

5. Softwareschutz

In diesem Kapitel möchte ich Ihnen einige Tips & Tricks zeigen, wie Sie Ihre Programme vor unautorisierten Benutzern schützen können. Es ist an dieser Stelle unmöglich, Ihnen einen hundertprozentigen Schutz anzubieten, denn den gibt es noch nicht. Der Sinn dieses Kapitels liegt auch nicht darin, den ausgeklügeltsten Schutz anzubieten, da das nicht der Sinn dieses Buches ist. Ich möchte Ihnen vielmehr einige kleine Tricks verraten, die jedoch recht wirksam sind. Benutzen Sie dann noch manche Tricks in Kombinationen, haben Sie so einen ausreichenden Schutz für Ihre Programme. Außerdem sind alle Tricks so erklärt, daß Sie die Funktionen verstehen können und so mit etwas Geduld noch mehr Tricks zum Softwareschutz herausfinden können.

5.1 Listen ohne Zeilennummern

Durch das Verändern der Adresse 22 kann man recht eigenartige Effekte erzielen. Tippen Sie bitte einmal ein kurzes Programm ein, und geben Sie dann

```
POKE 22,35
```

ein. Steht nach Drücken der RETURN-Taste der Cursor immer noch in der Zeile, so drücken Sie bitte noch einmal RETURN. Wenn Sie nun das Programm auflisten, werden die Zeilennummern unterschlagen. Sie können das Programm jedoch ganz normal laufen lassen. Es gibt jedoch eine Einschränkung: PRINT-Befehle werden nicht mehr ausgeführt. Der Ausgangszustand wird wieder durch das Erzeugen eines SYNTAX ERRORs oder durch den folgenden Befehl wiederhergestellt.

```
POKE 22,25
```

Wollen Sie das Programm auf den Drucker ausgeben, so müssen Sie die folgenden Befehlszeilen eingeben. Dabei ist jedoch zu beachten, daß die letzte Zeile unterschlagen wird.

```
OPEN 1,4,1
CMD 1
LIST
```

Durch Eingabe von

```
PRINT# 1
```

können Sie auch noch die letzte Zeile ausdrucken. Durch den Befehl

```
POKE 22,32
```

werden die Zeilennummern beim Auflisten unterschlagen, und nach der letzten Zeile werden einige Grafikzeichen ausgegeben. Der Befehl

```
POKE 22,25
```

macht dieses wieder rückgängig. Sollte einmal etwas Unerwartetes auftreten, während Sie den Wert der Speicherstelle 22 geändert haben, so liegt das in den meisten Fällen an der Manipulation dieser Speicherstelle. Schreiben Sie dann wieder den Normalwert (25) in die Adresse 22. Zu bemerken ist noch, daß ein Starten des Programmes den Normalzustand herstellt. Auch das Abändern des Programmlistings stellt den Normalzustand her.

5.2 Listschutz durch REM

Sie können Ihre Programme auch so ändern, daß einzelne Zeilen oder gar das ganze Programm nicht mehr aufgelistet werden. Dazu ist ein Befehl sehr wichtig, der sonst nichts bewirkt: der REM-Befehl. Der Vorteil der REM-Anweisung liegt ja bekanntlich darin, daß sie während des Programmablaufes ignoriert wird. Dadurch bricht das Programm im folgenden Beispiel auch nicht ab, wenn Sie es starten, denn dann wird der REM-Befehl nicht ausgeführt. Wenn Sie es auflisten, wird die REM-Anweisung jedoch nicht ignoriert, wodurch der Schutz entsteht. Geben Sie bitte einmal folgendes Programm ein:

```
10 PRINT "GESCHÜTZT"
20 REM <SHIFT/L>
30 PRINT "WEG"
```

Wenn Sie nun das Programm auflisten, so erscheint nach Zeile 20 ein

```
SYNTAX ERROR
```

und das Auflisten wird abgebrochen. Dieser Trick ist leider recht einfach zu durchschauen. Kombiniert man dies jedoch mit anderen Tricks, so ist er doch ganz wirksam. Setzen Sie an den Anfang des Programmes zehn solcher REM-Zeilen, so kann das Entfernen dieser Zeilen für den Unbefugten zu einer Tortur werden, da er nicht weiß, wie viele solche Zeilen noch kommen, wird er vielleicht aufhören, Ihr Programm zu knacken.

Wollen Sie nur wichtige Zeilen, nicht jedoch das ganze Programm schützen, so ist die oben genannte Methode ungeeignet, da durch diesen Schutz das Listen abgebrochen wird. Um nur eine Zeile zu schützen, geben Sie bitte einmal folgende Zeile ein:

```
10 PRINT "WEG" : REM ""
20 PRINT "DA"
```

Fahren Sie nun bitte mit dem Cursor auf das zweite Anführungszeichen der REM-Anweisung, und drücken Sie bitte 22mal die INSERT-Taste. Drücken Sie danach bitte 22mal die DELETE-Taste. Es müßten daraufhin reverse "T"s erscheinen. Haben Sie dies getan, so drücken Sie bitte RETURN. Wenn Sie nun das Programm auflisten, so erscheint kurz die zu schützende Zeile, doch direkt danach wird sie wieder gelöscht, und es wird mit der nächsten Zeile fortgefahren.

Der Trick funktioniert folgendermaßen. Das reverse "T" ist ein Steuerzeichen und steht für DELETE. Wird nun Zeile 10 aufgelistet, so erscheint erst der Anfang der Zeile. Wird jedoch dann der REM-Befehl aufgelistet, so wird das Steuerzeichen "DELETE" genau 22mal ausgeführt, es werden also die letzten 22

Buchstaben gelöscht. Da dies der Print-Befehl ist, wird er direkt nach dem Auflisten wieder gelöscht.

Sie sollten diesen Trick jedoch nicht bei langen Zeilen anwenden, denn je länger die Zeile ist, desto länger ist sie auf dem Bildschirm zu sehen, was sich durch ein kurzes Aufblitzen verrät. Am besten lassen sich so kurze Formeln und ähnliches schützen. Wenn diese relativ kurz sind, so sind sie beim Listen nicht zu erkennen. Es ist noch sinnvoll, die Zeilen durchdacht zu numerieren:

Haben Sie z.B. eine Zeilenschrittweite von 10 gewählt, also Ihre Zeilen immer in Zehnerschritten durchnumeriert, so kann es leicht auffallen, wenn eine Zeile - die geschützte also - fehlt. Geben Sie deshalb in diesem Fall einer geschützen Zeile eine 'krumme' Zeilennummer, also eine Endziffer zwischen 1 und 9, z.B. 45. Denn dort wird dann nicht erwartet, daß sich zwischen Zeile 40 und 50 eine geschützte Zeile befindet.

Einen weiteren Trick möchte ich Ihnen nicht vorenthalten. Wieder helfen uns dabei Steuerzeichen, nämlich das für den Wagenrücklauf und das für CURSOR-UP. Ziel dieses Schutzes ist es, den Cursor an den Anfang der Zeile zu setzen, damit diese direkt von der nächsten Zeile überschrieben wird. Geben Sie dazu einmal folgendes Beispiel ein:

```
10 PRINT "HALLO":REM" Nichts zu sehen!!!!!"
```

Fahren Sie nun mit dem Cursor hinter das erste Anführungszeichen des REM-Befehls, und drücken Sie <CRTL/9>. Diese Tastenkombination schaltet den Revers-mode an. Drücken Sie nun bitte <SHIFT/M> und dann <SHIFT/Q>. Daraufhin erscheint ein reverser Querstrich und ein reverser Kreis. Drücken Sie nun bitte noch <CRTL/0>, womit Sie den Revers-Mode abschalten. Bestätigen Sie Ihre Eingabe durch <RETURN>. Wenn Sie nun das Programm auflisten, so erscheint nur der Satz:

```
Nichts zu sehen!!!
```

Wie funktioniert dieser Trick? Nun, der reverse Schrägstrich ist ein Steuerzeichen für den Wagenrücklauf, was dem Drücken der RETURN-Taste gleichkommt. Da sich nun aber der Cursor unter der eben ausgegebenen Zeile befindet, muß er noch um eine Zeile nach oben bewegt werden. Das geschieht mit dem zweiten Steuerzeichen. Nun steht der Cursor wieder auf dem Anfang der zuletzt ausgebenen Zeile, welche nun durch den Text überschrieben wird. Der nächste Trick, den ich Ihnen vorstellen möchte, benutzt das Steuerzeichen <CLEAR/HOME>. Geben Sie bitte folgendes, etwas längere Programm ein!

```
10 PRINT "HEY" : REM " <CLR/HOME>"
20 A=15 : REM " <CLR/HOME>"
30 B=10 : REM " <CLR/HOME>"
40 C = A*B : REM " <CLR/HOME>"
50 PRINT C : REM " <CLR/HOME>"
60 A = A+1 : B = B+2 : REM " <CLR/HOME>"
70 IF B < A THEN 40 : REM " <CLR/HOME>"
80 END : REM " <CLR/HOME>"
```

Es ist wichtig, daß Sie zwischen dem Anführungszeichen der REM-Anweisung und dem reversen Herzen ein Leerzeichen lassen. Fahren Sie nun mit dem Cursor auf dieses Leerzeichen, schalten Sie den Revers-Mode an (<CRTL/9>), drücken Sie <SHIFT/M>, und schalten Sie den Revers-Mode wieder ab (<CRTL/0>). Daraufhin müßte ein reverser Schrägstrich erscheinen. Verfahren Sie nun genauso in den nächsten Zeilen.

Wenn Sie das Programm dann auflisten, werden Sie nicht viel lesen können, denn nach jeder Zeile wird der Bildschirm gelöscht und der Cursor in die linke obere Ecke gesetzt. Es überschreiben sich wieder alle Zeilen. Dieser Trick ist nur für kürzere Textpassagen passabel, da es doch sehr aufwendig ist, die Steuerzeichen hinter jede Zeile zu setzen.

Wußten Sie schon, daß man sein Programm auch durch Farbe schützen kann? Nein - dann lesen bitte weiter! Das Ziel des nächsten Tricks ist es, nach einer REM-Anweisung die Schriftfarbe in dunkelblau, also in die Hintergrundfarbe zu wechseln, da dann nichts mehr zu lesen ist. Geben Sie dazu bitte folgendes Programm ein:

```
10 PRINT "HALLO" : REM " <CRTL/7>"
20 PRINT
30 PRINT
```

Fahren Sie nun mit dem Cursor auf das erste Leerzeichen hinter den Anführungsstrichen der REM-Anweisung. Schalten Sie wieder den Revers-Mode ein (<CRTL/9>), und drücken Sie dann <SHIFT/M>. Schalten Sie nun den Revers-Mode aus (<CRTL/0>). Wenn Sie das Programm nun auflisten, so erscheint die erste Zeile, doch danach ist Schluß. Da auch nach dem Auflisten die Schriftfarbe immer noch dunkelblau ist, macht es den Eindruck, als ob der Rechner abgestürzt sei. Es ist selbstverständlich auch möglich, andere Farben als dunkelblau anzugeben, doch sind dann die Zeilen noch sichtbar.

In diesem Kapitel habe ich Ihnen gezeigt, wie man durch Kombination von Steuerzeichen und REM-Befehlen seine Programme vor unautorisierten Benutzern schützen kann. Ich habe Ihnen hier nur Beispiele für verschiedene Steuerzeichen und deren Realisierung in REM-Befehlen gegeben. Auch hier gilt wieder: Probieren geht über Studieren!

5.3 Schutz durch Zeichenkombinationen

Die nächste Methode, die ich Ihnen zum Programmschutz vorstellen möchte, ist ein Programm, das das zu schützende Programm abändert. Alle Zeilen, die geschützt werden sollen, sollen durch drei Doppelpunkte gekennzeichnet werden. Wird nun das Programm aufgerufen, so werden alle gekennzeichneten Zeilen vor dem Einsehen geschützt. Geben Sie dazu einmal folgendes Programm ein:

```
10 PRINT "DA"
20 ::: PRINT "NICHT DA"
30 PRINT "DA"
```

Es soll also in diesem Programm die Zeile 20 geschützt werden, da sie durch drei Doppelpunkte gekennzeichnet ist. Nun kommt die eigentliche Routine, die unser Programm nach drei Doppel-

punkten durchsuchen und anschließend diese Zeilen schützen soll. Geben Sie das diesmal etwas längere Programm bitte sorgfältig ein:

```
10000 FOR A = PEEK (43) + PEEK (44)*256 TO PEEK (45) + PEEK
(46)*256
10001 IF PEEK (A)=58 THEN 10010
10002 NEXT A
10003 END
10010 IF PEEK (A+1)<>58 THEN 10002
10011 IF PEEK (A+2)=58 THEN POKE A,0
10012 GOTO 10002
```

Speichern Sie diese Routine bitte erst ab, bevor Sie sie starten, denn falls Sie etwas falsch eingetippt haben sollten, so könnte dadurch unsere Routine zerstört werden. Rufen Sie nun die Routine durch

```
GOTO 10000
```

auf. Nach kurzer Zeit meldet sich der Computer wieder durch die READY-Meldung. Wenn Sie nun das Programm auflisten, so wird Zeile 20 einfach übergangen. Um Ihnen den Trick zu erklären, muß ich einen kleinen Ausflug in die BASIC-Welt des C64 machen: Da der BASIC-Interpreter auch wissen muß, wann eine Programmzeile zu Ende ist, gibt es dafür eine Kennung, nämlich das sogenannte Nullbyte, also das Zeichen mit dem Character-Code Null. Erkennt der Rechner dieses Byte, so 'glaubt' er, daß die Zeile dort zu Ende sei, und listet die nächste Zeile auf.

Die oben aufgelistete Routine durchsucht also das BASIC-Programm nach der Kennung, und POKEd dann an die Stelle des ersten Doppelpunktes eine Null. Listet der Rechner nun das Programm auf, so erkennt er direkt am Anfang der Zeile 20 ein Nullbyte. Für ihn ist die Zeile nun zu Ende, weshalb er die nächste Zeile, also in diesem Falle Zeile 30, auflistet. Wollen Sie nicht immer die Routine mit abspeichern, so sollten Sie diese nach dem Gebrauch aus dem Speicher löschen.

5.4 Umbiegen des List-Vektors

Bisher haben wir immer das Auflisten eines BASIC-Programmes beeinflußt, nicht jedoch das Listen selbst verhindert. Daß man das Listen auch abschalten kann, werde ich Ihnen nun zeigen. Geben Sie bitte einmal folgendes ein:

```
POKE 774,131
POKE 775,246
```

Wenn Sie nun Ihr Programm auflisten wollen, so passiert nichts, der Computer meldet sich mit einer simplen READY-Meldung. Wie funktioniert das? Nun, wird die List-Routine durch Eingabe von LIST aufgerufen, so verzweigt der Rechner intern zu einer Routine, die Tokens in Klartext umwandelt. Der Rechner verzweigt jedoch nicht direkt zu der Routine, sondern er holt sich erst die Adresse, zu der er springen muß, aus zwei Speicherzellen, dem sogenannten LIST-Vektor. Diese zwei Speicherzellen enthalten normalerweise die Werte 26 und 167, der Zeiger zeigt also auf die Adresse 42778, wo auch die List-Routine beginnt.

Der Zeiger der Speicherstelle 774 und 775 wurde so geändert, daß er nun auf den Maschinensprachebefehl RTS zeigt, was einen Rücksprung zum BASIC bedeutet. Wird nun LIST eingegeben, so holt sich der Rechner die Adresse, die er anspringen muß, aus diesen beiden Speicherzellen. Danach springt er diese an und führt das dortige Programm auch aus. Da dort aber nur der Befehl RTS steht, wird direkt wieder zum BASIC zurückgesprungen, und die READY-Meldung erscheint. Wie Sie sehen, ist es möglich, jeden beliebigen Wert in diese beiden Speicherzellen zu schreiben, womit die LIST-Routine beliebig abgeändert werden kann. Hier nun ein weiteres Beispiel:

```
POKE 774,226
POKE 775,252
```

Wenn Sie die beiden Werte umrechnen, so erhalten Sie die Adresse 64738, die Ihnen vielleicht bekannt vorkommt. Die Zeiger zeigen auf die Reset-Routine. Bei der Eingabe von LIST wird also ein Reset ausgeführt. Am effektivsten ist es jedoch,

wenn man den Rechner zu einem Befehl verzweigen läßt, bei dem der Rechner abstürzt. Dies geschieht, wenn man in die Speicherstellen 774/775 die Werte 225/171 schreibt. Geben Sie einmal folgendes ein:

```
POKE 774,225
POKE 775,171
```

Hiermit verzweigt der Rechner nun nach 44001, wo er einen sogenannten Illegal Opcode ausführt, was zum Absturz des Systems führt. Dem unbefugten Benutzer bleibt nun nichts anderes übrig, als den Rechner auszuschalten oder einen Reset durchzuführen. Einen Nachteil hat die Methode jedoch: Das Programm muß erst einmal gestartet werden, damit die Pokes überhaupt den List-Vektor verändern können. Wenn Sie Ihr Programm jedoch mit einem Autostart versehen, so ist dieser Trick schon recht schwer zu durchschauen.

5.5 Abschalten gefährlicher Tasten

Bisher wurde vorausgesetzt, daß das Programm nicht durch Drücken irgendwelcher Tasten abgebrochen wurde. So löst z.B. das Drücken der Tasten <RUN/STOP-RESTORE> einen Interrupt aus, und das gerade laufende Programm wird abgebrochen. Wurde vorher jedoch noch kein List-Schutz aktiviert (z.B. durch Umbiegen des List-Vektors), so ist das Programmlisting einsehbar.

Es müssen also solche 'gefährlichen' Tasten abgeschaltet werden können. Zuerst gilt es, die STOP-Taste auszuschalten, da diese unter anderem dafür gedacht ist, ein laufendes Programm zu unterbrechen. Geben Sie bitte folgende Befehle ein:

```
POKE 788,52
```

Ab nun wird ein Programm nicht mehr beim Drücken der STOP-Taste abgebrochen. Dazu ist wieder etwas Hintergrundwissen nötig. Der C64 ruft jede 50stel Sekunde eine Routine, die sogenannte Interrupt-Routine, auf. Ein Interrupt (engl.: Unterbrechung) ist - wie der Name schon sagt - eine Unterbrechung

des normalen Ablaufes, in der einige Dinge erledigt werden. So wird z.B. die interne Uhr aktualisiert, und ähnliche Zustände werden kontrolliert. Auch beim Interrupt springt der Rechner über einen Zeiger zu einer Routine, die dann ausgeführt wird. Die erste Aufgabe der Interrupt-Routine ist es zu überprüfen, ob die STOP-Taste gedrückt wurde und in dem Fall das gerade laufende BASIC-Programm zu unterbrechen.

Diese Überprüfung muß also unterbunden werden. Dies geschieht dadurch, daß das High-Byte des Interrupt-Zeigers um drei erhöht wird, wodurch nun die Überprüfung der STOP-Taste quasi übersprungen wird. Deshalb muß auch nur ein Teil des Zeigers geändert werden. Den Normalwert 788 können Sie wieder durch

```
POKE 788,49
```

in die Speicherstelle 788 schreiben. Hier nun noch ein kleiner Hinweis zur STOP-Taste, der jedoch nicht viel mit Softwareschutz zu tun hat: Sie können das Drücken der STOP-Taste, sofern sie abgeschaltet ist, auch per Programm abfragen, denn sie hat den Character-Code 3. Durch

```
10 GET A$
20 IF ASC (A$) = 3 THEN ...
```

geschieht dies. Des weiteren kann man die RESTORE-Taste als gefährlich einstufen, denn bekanntlich wird ein Programm ja durch das Drücken der Tasten <RUN/STOP-RESTORE> abgebrochen. In diesem Falle wird auch ein Interrupt ausgeführt, der sogenannte NMI (Non Mascerable Interrupt). Auch hier gibt es einen Zeiger, der auf diese Routine zeigt. Im Falle des Drückens der RUN/STOP-RESTORE-Tasten holt sich also der Rechner die Ansprungsadresse aus der Speicherzelle 792/793 und verzweigt dann zu der angegebenen Routine. Um dem Programmabbruch durch <RESTORE> vorzubeugen, muß der Zeiger so geändert werden, daß die Interrupt-Routine ganz übersprungen wird. Dies geschieht durch folgenden Poke:

```
POKE 792,193
```

Wollen Sie den Abbruch durch <RESTORE> wieder zulassen, so müssen Sie den Wert 71 in die Speicherstelle 193 schreiben. Es ist natürlich auch möglich, daß ein Reset ausgelöst wird, sobald <RESTORE> gedrückt wird. Dazu müssen Sie den Zeiger so 'umpolen', daß er auf die Reset-Routine zeigt:

```
POKE 792,226
POKE 793,252
```

Hier sind wieder alle Werte zugelassen. Sofern Sie mit Maschinensprache vertraut sind, können Sie auch eigene Routinen schreiben, die dann beim Drücken einer der Tasten angesprungen werden. Wollen Sie einmal die Tastenkombination <RUN/STOP-RESTORE> ausschalten, so können Sie dies durch Ihr eben erworbenes Wissen verwirklichen. Es geht jedoch auch einfacher. Wird nämlich <RUN/STOP-RESTORE> gedrückt, so verzweigt der Rechner über den STOP-Vektor, der normalerweise auf die Adresse 63213 zeigt. Um dem Programmabbruch durch <RUN/STOP-RESTORE> vorzubeugen, muß der Zeiger wieder so abgeändert werden, daß die Abfrage nach der STOP-Taste überspungen wird. Dies geschieht durch

```
POKE 808,254
```

Der normale Wert der Speicherzelle 808 beträgt 237. Diese Tricks sind leider nicht hundertprozentig, da ein Programm immer noch durch einen Hardware-Reset unterbrochen werden kann. Da das aber ein Extremfall ist (außerdem ist das Programm dann nicht mehr auflistbar), können wir diesen Fall hier ruhig vergessen.

5.6 Selbstschützende Programme

Soll ein Programm vor dem Auflisten geschützt werden, so empfiehlt es sich, nach dessen Beendigung den BASIC-Speicher zu löschen oder dieses zumindest zu simulieren. Wie Ihnen vielleicht bekannt ist, löscht der NEW-Befehl den BASIC-Speicher. Es ist daher die einfachste Methode, ein Programm mit dem Befehl NEW zu beenden. Hierzu ein kleines Beispielprogramm:

```
10 PRINT "NOCH IST ES DA!!"
20 PRINT "GLEICH IST ES WEG"
30 NEW
```

Starten Sie das Programm, so ist es nach dem Programmablauf auch nicht mehr auflistbar. Ein zweiter Weg, ein Programm vor dem Auflisten zu schützen, besteht darin, die ersten Bytes des Programmes 'umzupoken'. Erkennt der Interpreter nämlich zwei Nullbytes in einem Programm, so bedeutet dies das Ende des Programmes. Poken wir nun also zwei Nullen an die Anfangsadressen des Programmes, so ist es nicht mehr auflistbar. Hierzu ein Beispiel:

```
10 PRINT "HALLO"
20 POKE 2049,0
30 POKE 2050,0
```

Wollen Sie nun nach dem Programmablauf das Programm auflisten, so gibt der Rechner nur ein einfaches READY aus, denn er erkennt zuerst zwei Nullbytes, also bricht er das Listen ab. Der Nachteil dieser Methode liegt natürlich darin, daß Sie das Programm nicht mehr vernünftig starten können. Um das wieder zu erreichen, müssen Sie lediglich die richtigen Werte in die beiden Speicherstellen POKEn, die ganz von Ihrem Programm abhängen. Lassen Sie sich diese am besten direkt vor dem Programmstart durch

```
PRINT PEEK (2049)
PRINT PEEK (2050)
```

ausgeben, und verwahren Sie diese beiden Zahlen an einem sicheren Ort. Verstärkt wird dieser Schutz noch dadurch, daß Sie zwei weitere Bytes auf Null setzen. Sollte nämlich der Benutzer einmal auf die beiden Anfangswerte kommen, so gäbe es immer noch eine Abbruchstelle im Programm. Auch hier gilt: Merken Sie sich die Zahlen, damit Sie das Programm noch selber starten können. Diese eben genannten Tricks funktionieren nur nach einem Programmstart, denn sonst werden die Befehle ja nicht ausgeführt. Eine Kombination mit einem Autostart ist auch hier wieder recht nützlich.

5.7 Autostart

In den vorigen Kapiteln wurde beschrieben, wie man Programme gegen unbefugtes Einsehen oder Kopieren schützen kann. Doch was nützt der beste Listschutz, wenn dieser im Programm steht und erst aktiviert wird, wenn das Programm gestartet wurde, es nach dem Laden also auflistbar ist. Nun gibt es die Möglichkeit, ein Programm so auf Diskette abzulegen, daß es sich nach dem Laden automatisch sofort startet. Dies ist vorteilhaft, wenn Sie Ihr Programm von jemandem bedienen lassen wollen, der mit Computern nicht so vertraut ist, da man ihm nicht erklären muß, wie man das Programm startet.

Eine Methode will ich Ihnen nun vorstellen, die zwar nicht das Optimum an Komfort bietet, aber die einzige mir bekannte Methode ist, die ohne Maschinensprache nur mit reinem BASIC auskommt. Diese Methode kennen Sie eigentlich schon, wenngleich Sie es sich wohl nicht so bewußt gemacht haben. Bitte lesen Sie sich noch einmal die Ausführungen zum programmierten Direktmodus durch (Kap. 4.2.3). Genauso soll nun unser Autostart funktionieren.

Man schreibt den RUN-Befehl einfach auf den Bildschirm, in den Tastaturpuffer das Zeichen chr$(13) und speichert diesen Puffer zusammen mit dem Bildschirmspeicher-BASIC-Programm absolut ab. Der BASIC-Anfang (Speicherstellen 43 und 44) muß vor dem RUN wieder hochgesetzt werden. In Kapitel 3.5 ist beschrieben, wie man bestimmte Speicherbereiche direkt auf Diskette (oder Kassette) ablegt.

Schlagen Sie jetzt doch einfach mal das Buch zu und versuchen anhand der gegebenen Hinweise selbst einen einfachen Autostart zu programmieren. Sie lesen immer noch? Ich werde ihnen im folgenden Schritt für Schritt zeigen, wie man einen Autostart erzeugt.

- Laden Sie das zu bearbeitende BASIC-Programm in den Speicher.

- Geben Sie ein:

```
POKE 43,198     (Anfangsadresse des absoluten PRG-Files)
POKE 44,0
POKE 198,2:POKE 631,19:POKE632,13:?"<CLR/HOME>POKE 43,1:POKE
44,8:RUN":SAVE"AUTO",8,1
```

Dies ist nun wieder der programmierte Direktmodus, wobei die Pokes im PRINT-Befehl nötig sind, um die BASIC-Zeiger nach dem Laden wieder zu korrigieren. Die letzten Pokes sollten unbedingt mit dem Save-Befehl zusammen in einer Zeile eingegeben werden. Wenn Sie nun mit

```
LOAD "AUTO",8,1
```

das Programm laden, so erscheint zunächst die Zeile

```
POKE 43,1:POKE 44,8:RUN
```

auf dem Bildschirm, und nach Beendigung des Ladevorganges erscheint nicht etwa

```
READY
```

auch ein eingabebereiter Cursor nach dem Ladevorgang will sich nicht blicken lassen, es wird schlicht und einfach das Programm gestartet, in dessen ersten Zeilen nun die bekannten Pokes für List- und Stopschutz stehen. Sicher ist es nicht befriedigend, wenn man beim Laden die Programmzeile auf dem Bildschirm sieht.

Eine einfache Möglichkeit, dies abzuändern, besteht darin, weitere Zeichen in den Tastaturpuffer zu poken (bis zu 10 Zeichen sind 'legal' möglich, mit Tricks noch mehr). Diese weiteren Zeichen können Steuerzeichen sein, die z.B. die Zeichenfarbe auf dunkelblau setzen, so daß man sie normalerweise nicht mehr sehen kann. Hier würde ein einfacher

```
POKE 53281,0
```

reichen, um sich Klarheit über die Funktionsweise des Programmes zu verschaffen, ein Poke, den mancher Besitzer eines Scharz-Weiß-Fernsehers wegen der besseren Bildqualität ohnehin

routinemäßig beim Einschalten eingibt. Ein weiterer Nachteil dieser Methode ist, daß man während des Ladens mittels der Stop-Taste das Programm abbrechen kann und, je nach Geschick, den Ladevorgang so kurz vor Beendigung abbrechen kann, daß das Programm vollständig im Speicher zum Angucken bereitsteht.

Dies können Sie vermeiden, indem Sie schon vor dem SAVE-Befehl die Pokes zum Ausschalten der Stop-Taste eingeben, schließlich wird der gesamte Bereich zwischen 198 und dem BASIC-Anfang (2048) abgespeichert. Zwischen 198 und 2048 liegt der Bildschirmspeicher. Dies können Sie ausnutzen, indem Sie den PRINT-Befehl vor dem SAVE-Befehl um sinnvolle Texte erweitern. Dies wird dann während des Ladens als Titelbild angezeigt, falls Sie nicht über die schon erwähnte alte Betriebssystemversion verfügen.

5.8 Reset-Taster

In der Hardware des C64 ist es vorgesehen, daß durch Aktivieren einer bestimmten Leitung am Prozessor der C64 auf den Ausgangszustand zurückgesetzt wird - diese Leitung wird beim Anschalten des Rechners angesprochen, um einen definierten Einschaltzustand zu erreichen. In der Reset-Routine wird z.B. der Speicher gelöscht (genau wie bei NEW, die Renew-Routine aus Kapitel 8.3. funktioniert also), die Peripheriegeräte (Drucker, Floppy) werden zurückgesetzt, der VIC und SID initialisiert, die Einschaltmeldung ausgegeben und vieles mehr.

Diese Reset-Leitung wird beim C64 im USER-Port herausgeführt, so daß es möglich ist, von außen einen RESET auszulösen. Hiermit kommen Sie dann aus (fast) jedem Programm heraus, und nach einem RENEW (Kapitel 8.3) können Sie sich dann das Programm, sofern es ein BASIC-Programm war, anschauen. Langer Rede, kurzer Sinn, hier eine Bauanleitung für einen einfachen Reset-Taster (Kostenpunkt: unter DM 5,-).

Sie besorgen sich einen USERPORT-Stecker (z.B. bei Conrad-Electronic Klaus-Conrad-Str. 1, 8452 Hirschau, Best.-Nr. 74 06 91), einen Lötkolben und ein Stück Draht; wenn Sie es ganz professionell machen wollen, zusätzlich ein ebenfalls z.B. bei Conrad unter Best. Nr. 74 01 10 erhältliches passendes Gehäuse und einen kleinen Taster. Verbinden Sie nun die Pins 1 und 3 in der Abbildung mit dem Draht untereinander. Wenn Sie diesen Stecker nun kurz in den USERPORT stecken und sofort wieder herausziehen, so wird ein Reset ausgelöst.

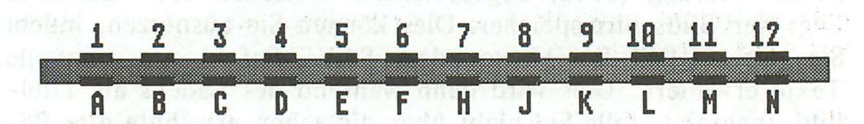


Abb. 5 User-Port

In der professionellen Ausführung nehmen Sie das Gehäuse, schrauben oben den Taster darauf, und verbinden einen Pol des Tasters mit Pin [1] und den anderen Pol mit Pin [3]. Diesen Taster können Sie dauernd auf dem USERPORT stecken lassen, bei Bedarf drücken Sie den Taster, und schon wird ein Reset ausgelöst. Bitte beachten Sie, daß in einigen alten Büchern und Zeitschriften noch die Methode des RESET über den Floppy-Port beschrieben wird. In neueren Geräten ist die RESET-Leitung im Floppy-Port nur noch als Ausgang und nicht mehr als Eingang geschaltet. Daher funktionieren RESET-Taster, die in den Floppy-Port gesteckt werden, nur bei alten C64.

5.9 Schutz vor RESET

Nun habe ich Ihnen hier eine Methode vorgestellt, alle bisherigen Tips hinsichtlich Softwareschutz zu umgehen, und muß daher auch zeigen, wie man den RESET-Taster softwaremäßig außer Gefecht setzen kann. Man bedient sich dabei der Tatsache, daß nach jedem Reset die Initialisierungsroutine aufgerufen

wird, die unter anderem prüft, ob ein Modul im Modulschacht steckt, um diesem dann die Kontrolle über den Prozessor zu übergeben.

Woran erkennt nun die Reset-Routine, ob ein Modul eingesteckt wurde? Ein Modul liegt für den Prozessor zunächst einmal wie ein normaler Speicherbaustein vor. Sog. Autostart-Module beginnen meist an der Adresse $8000=32768. Zur Kennzeichnung, daß dieses Modul nach dem Einschalten des Rechners sofort aktiviert werden soll, steht in der Adresse $8000 die Startadresse des Modulprogrammes und ab Adresse $8004 die Kennzeichnung CBM80 (CBM unbedingt groß schreiben, also die 5 Bytes $c3,$c2,$cd,$38,$30)

Dies können wir nun ausnutzen, indem wir in den normalen RAM-Speicher die Kennung CBM80 hineinpoken und so dem Betriebssystem vormachen, ein Modul sei angeschlossen. In die Speicherstelle $8000 setzen wir einen Zeiger auf ein abstürzendes Programm (oder gar das Hauptmenü unseres zu schützenden Programmes). Folgende Befehlsfolge am Anfang Ihres Programmes, bewirkt, daß durch einen Reset nicht der Rechner in den Ausgangszustand zurückkehrt, sondern abstürzt:

```
10 FOR T=32768 TO 32777
20 READ A:POKE T,A
30 NEXT T
40 DATA 9,128,0,0:  REM Adresse $8009, 2 dummy-Bytes
50 DATA 195,194,205,56,48: REM CBM80
60 DATA 2: REM abstürzender illegal Opcode
```

Wenn dieser Vorspann vor Ihrem Programm steht, ist es nicht mehr möglich, dieses durch einen Reset-Taster wie oben beschrieben zu knacken. Allerdings ist Ihr Programm selbst so noch nicht 100% geschützt, wenngleich schon kaum noch etwas passieren kann. Es gibt nämlich eine kompliziertere Ausführung des Reset-Tasters, welcher trotz Autostart-Simulation ein Programm unterbrechen kann. Wenngleich ich hier keine Bauanleitung liefern will, da dieses Buch primär Softwaretips bieten soll, gehe ich doch kurz auf das Prinzip ein:

Im Expansionsport sind verschiedene Prozessorleitungen herausgeführt, u.a. auch solche, die die interne Speicheraufteilung betreffen. Schließlich muß ein Modul ja auch die RAMs ab $8000 ausblenden können. Genau das macht sich der erweiterte Reset-Taster zunutze, indem das RAM ausgeblendet wird, die RESET-Routine nicht im Speicher, sondern in einem nicht vorhandenen Modul nach CBM80 sucht, es dort nicht findet und normal weiterarbeitet. Da dieser Reset-Taster nicht sehr weit verbreitet ist, bieten die oben beschriebenen Tips aber einen ausreichend guten Programmschutz.

6. Grafik auf dem C64

Dieses Kapitel soll Ihnen nun zeigen, wie Sie Farbe in Ihre Programme kriegen, wie Sie den Grafikmodus programmieren und wie Sie den Zeichensatz des C64 ändern können. Durch solche Tricks, die oft nur Details ausmachen, werden Ihre Programme nicht nur übersichtlicher, sondern sie sehen auch professioneller aus, denn welches Programm besitzt schon nur einen Strich als Cursor? Außerdem werden wir Ihnen zeigen, wie man im hochauflösenden Modus Punkte setzt, was vom BASICV2 überhaupt nicht unterstützt wird.

6.1 Verändern der Zeichenfarbe

Wußten Sie schon, daß die Zeichenfarbe beim C64 nicht immer hellblau sein muß? Durch Kombination von zwei Tasten läßt sich die aktuelle Zeichenfarbe ändern. Insgesamt stehen Ihnen bis zu 16 verschiedene Farben zu Verfügung, die Sie auch im Direktmodus einstellen können. Um die Farbe zu ändern, müssen Sie lediglich die CRTL-Taste bzw. die Commodore-Taste mit einer der Zahlen 1-8 drücken. Drücken Sie bitte einmal die CRTL-Taste mit der 1. Daraufhin wechselt der Cursor seine Farbe. Er ist nun schwarz.

Sollten Sie nun etwas experimentiert haben, und war der Cursor plötzlich gar nicht mehr da, so haben Sie die Kombination <CRTRL-7> gedrückt, denn dann schaltet die Farbe auf dunkelblau, weshalb der Cursor nicht mehr zu erkennen ist. Am Ende dieses Kapitels habe ich Ihnen alle möglichen Farben mit deren Tastenkombinationen und Zahlen tabellarisch aufgeführt. Was ist, wenn nun in einem Programm die Zeichenfarbe wechseln soll. Auch dazu gibt es zwei Methoden: Die erste Methode besteht darin, die Zeichenfarbe mittels PRINT-Befehl und einem Steuerzeichen zu ändern. Der Befehl

```
PRINT CHR$(158)
```

bewirkt z.B., daß die Zeichenfarbe von nun an gelb ist. Dies funktioniert jedoch nur mit den 8 Grundfarben, da nur für sie Steuercodes reserviert sind. Die zweite Methode besteht darin, in die Speicherstelle 646 einen Wert zu POKEn. In dieser Speicherstelle ist nämlich die aktuelle Zeichenfarbe gespeichert, und wenn dieser Wert nun geändert wird, so ändert sich auch die Zeichenfarbe. Geben Sie bitte einmal folgendes ein:

```
POKE 646,1
```

Daraufhin müßte der Cursor nun in weißer Farbe erscheinen. Vielleicht werden Sie sich nun überlegt haben, daß es eigentlich mehr als 16 Farben geben müßte, da eine Speicherstelle ja bekanntlich bis zu 256 verschiedene Werte annehmen kann. Doch da muß ich Sie leider enttäuschen. Der C64 kennt 'nur' die 16 Farben. POKEn Sie einen Wert über 16 in die Speicherzelle 646, so wiederholen sich die Farben nur. Der Wert 32 bewirkt also das gleiche wie der Wert 0: Die Farbe wird schwarz. Für Eingeweihte: nur die unteren 4 Bits beeinflussen die Zeichenfarbe, die oberen 4 Bits werden ignoriert.

Da wir schon gerade dabei sind, so kann ich Ihnen noch kurz zeigen, wie Sie die Rahmen und die Hintergrundfarbe verändern können. Auch dafür gibt es zwei Speicherstellen. Um die Hintergrundfarbe zu ändern, müssen Sie in Speicherstelle 53281 einen entsprechenden Farbwert POKEn, während ein POKE in die Speicherstelle 53280 die Rahmenfarbe ändert. Die Befehle

```
POKE 53280,0   : REM  Rahmenfarbe schwarz
POKE 53281,0   : REM  Hintergrund schwarz
POKE 646,5     : REM  Zeichenfarbe grün
```

lassen das Bild doch schon viel professioneller erscheinen, oder? Hier nun die Tabelle der verschiedenen Farben, deren Tastenkombinationen und deren Nummer:

Schwarz	CRTL-1	0	CHR$(144)
Weiß	CRTL-2	1	CHR$(5)
Rot	CRTL-3	2	CHR$(28)
Türkis	CRTL-4	3	CHR$(156)
Violett	CRTL-5	4	CHR$(159)

Grün	CRTL-6	5	CHR$(30)
Blau	CRTL-7	6	CHR$(31)
Gelb	CRTL-8	7	CHR$(158)
Orange	Commodore-1	8	
Braun	Commodore-2	9	
Hellrot	Commodore-3	10	
Grau1	Commodore-4	11	
Grau2	Commodore-5	12	
Hellgrün	Commodore-6	13	
Hellblau	Commodore-7	14	
Grau3	Commodore-8	15	

6.2 Bestimmen der Hintergrundfarbe eines Zeichens

Wußten Sie schon, daß man die Hintergrundfarbe eines einzelnen Zeichens bestimmen und verändern kann? Der C64 besitzt ab Adresse 55296 bis Adresse 56319 das sogenannte Farb-Ram. Es umfaßt 1024 Byte. Hier ist für jede Bildschirmkoordinate die Hintergrundfarbe gespeichert. Sie können die Farben nun durch POKEn von entsprechenden Farbwerten abändern. Die Adresse für die entsprechende Speicherstelle abhängig von der Koordinate erhalten Sie durch folgenden Befehl:

```
POKE 55296 + ZEILE*40 + SPALTE,FARBWERT
```

Wollen Sie z.B. die Hintergrundfarbe der Koordinaten 10/2 in schwarz umändern, so ist folgender Poke nötig:

```
POKE 55296 + 2*40 + 10,0
```

Hier nun eine kleine Routine, die die Hintergrundfarbe einmal auf andere Weise ändert:

```
10 FARBRAM=55296
20 FOR A=0 TO 3
30 FOR B=0 TO 999 STEP 4
40 POKE FARBRAM+B+A,1
50 NEXT B
60 NEXT A
```

6.3 Setzen der Hintergrundfarbe im Textmodus

Im vorherigen Kapitel habe ich Ihnen gezeigt, wie Sie die Farbe der Schrift ändern können und wie die Hintergrundfarbe geändert werden kann. Es war jedoch nicht möglich, die Hintergrundfarbe eines einzelnen Zeichens zu verändern. Um es vorwegzunehmen: Es ist auch nicht möglich, die Hintergrundfarbe eines Zeichens im normalen Modus durch einen Poke zu verändern.

Man kann sich jedoch helfen, indem man den Modus wechselt, was jedoch etwas aufwendiger ist. Da dies jedoch die einzige Methode ist, die Hintergrundfarbe eines Zeichens zu verändern, möchte ich Ihnen in diesem Kapitel verraten. Wie schon erwähnt, besitzt der C64 einen erweiterten Textmodus. Aktivieren können Sie diesen Modus durch

```
POKE 53265,PEEK (53265) OR 64
```

Durch diesen Befehl wird das 6. Bit der Adresse 53265 (Register 17 des VIC) gesetzt, wodurch der erweiterte Textmodus eingeschaltet wird. Nun müssen wir natürlich noch die Farben festlegen, die der Hintergrund eines Zeichens haben soll. Dies geschieht in den Registern 33 bis 36 des VIC. Das Register 33 ist die Adresse 53281, das Register 34 ist die Adresse 34 usw. Sie haben nun insgesamt 4 Hintergrundfarben zur Verfügung. Jetzt kommt natürlich die Frage auf, wie man die einzelnen Farben setzt, denn der normale PRINT-Befehl zaubert keine besondere Hintergrundfarbe auf den Bildschirm. Geben Sie bitte einmal folgendes ein:

```
POKE 53282,10
```

Geben Sie nun bitte ein paar geshiftete Zeichen ein, drücken Sie also <SHIFT> und ein Zeichen gleichzeitig. Sie werden merken, daß diese Zeichen nun nicht mehr die normale Hintergrundfarbe haben, sondern daß nun einen hellroten Hintergrund haben. Das sieht doch besser aus, oder? Geben Sie nun wieder einige Zei-

chen ohne <SHIFT> ein. Die Hintergrundfarbe dieser Zeichen wird wieder wie vorher sein. Geben Sie nun bitte folgenden Poke ein:

```
POKE 53282,5
```

Nun haben geshiftete Zeichen nicht mehr die Hintergrundfarbe hellrot, sondern grün. Noch einmal zur Verdeutlichung: Haben wir den erweiterten Textmodus eingeschaltet, so haben 'normal' eingebene Zeichen die Hintergrundfarbe der Adresse 53281, werden Zeichen mit <SHIFT> eingegeben, so erhalten Sie ihre Hintergrundfarbe aus der Adresse 53282.

Vielleicht haben Sie nun schon experimentiert und sind von alleine drauf gekommen: Da wir ja vier verschiedene Hintergrundfarben zur Verfügung haben und wir es bis jetzt erst auf zwei brachten, muß es noch andere Möglichkeiten geben. Geben Sie bitte folgenden Poke ein:

```
POKE 53283,7
```

Drücken Sie nun bitte CRTL-9, wodurch der Revers-Mode eingeschaltet wird. Wenn Sie nun ein paar Zeichen eingeben, so werden Sie merken, daß diese nicht mehr die normale Hintergrundfarbe - nämlich dunkelblau - haben, sondern daß deren Hintergrund nun gelb ist. Nun wird also die Hintergrundfarbe durch den Wert der Adresse 53283 bestimmt.

Die vierte und auch letzte Hintergrundfarbe erreichen wir, indem wir geshiftete Zeichen im Revers-Mode eingeben, also durch Kombination der beiden zuletzt genannten Methoden. Wie nicht schwer zu erraten ist, ist die Adresse 53284 für die Hintergrundfarbe dieser Zeichen zuständig. Setzen Sie also bitte einmal zur Verdeutlichung die Hintergrundfarbe der geshifteten Zeichen im Revers-Mode auf schwarz. Dies geschieht durch

```
POKE 53284,0
```

Schalten Sie nun in den Revers-Mode (sofern nicht schon geschehen), und geben Sie einige Zeichen beim gleichzeitigen Drücken der SHIFT-Taste ein. Diese Zeichen sollten nun die

Hintergrundfarbe schwarz haben. Hier nun noch einmal die tabellarische Auflistung der verschiedenen Hintergrundfarben:

Modus	**zuständige Adresse**
normal	53281
shift	53282
revers	53283
shift+revers	53284

Wie so vieles hat auch der erweiterte Grafikmodus seinen Nachteil. Es stehen einem leider nur noch 64 verschiedene Zeichen zur Verfügung, was auch einleuchtet, denn im erweiterten Textmodus dienen die beiden Bits 6 und 7 als Zeiger auf die Hintergrundfarbe, womit nur noch 6 Bits für die Zeichen übrigbleiben. Wollen Sie also unbedingt Grafikzeichen darstellen, so bleibt Ihnen nichts anderes übrig, als im normalen Textmodus zu arbeiten.

Sie können im erweiterten Textmodus alle Zeichen, die den Character-Code 1-65 haben, darstellen. Genau gesagt heißt das: alle Buchstaben, Zahlen sowie die allgemein gängigen Zeichen (%, º, (, *, ...). Normalerweise werden diese Zeichen jedoch teilweise durch Kombination mit der SHIFT-Taste erreicht. So bringt man z.B. durch Drücken der Tasten <SHIFT-1> das Ausrufungszeichen auf den Bildschirm.

Nun kommt natürlich die Frage auf, ob das Drücken von SHIFT und einer anderen Taste nicht eine andere Hintergrundfarbe als die der Speicherstelle 53281 erzeugt, denn drückt man SHIFT und eine andere Taste, so ist ja die Adresse 53282 für die Hintergrundfarbe zuständig. Man könnte meinen, daß es hier nun ein großes Durcheinander geben könnte.

Dies ist jedoch nicht der Fall. Möchten Sie z.B. im Revers-Mode das Dollar-Zeichen auf den Bildschirm bringen, so erreichen Sie dies ganz normal durch Drücken der Tasten <SHIFT>-4. Sollen jetzt aber die Farbregister 53283 oder 53284 für die Hintergrundfarbe des Dollar-Zeichens verantwortlich sein, so erreichen Sie das durch Drücken einer speziellen Tastenkombination, die nur im erweiterten Textmodus gilt. Um das Dollar-Zeichen mit

der Hintergrundfarbe aus dem Register 53283 darzustellen, müssen Sie die Tastenkombination <COMMODORE>-<KLAMMERAFFE> drücken. Sie können es also nicht mehr durch Drücken von <SHIFT>-4 erreichen, da sonst die Hintergrundfarbe durch den Wert der Adresse 53282 bestimmt würde.

Fassen wir noch einmal zusammen, wie Sie Zahlen und andere gängige Zeichen im erweiterten Textmodus auf den Bildschirm bringen können. Sollen die Adressen 53281 oder 53283 die Hintergrundfarbe bestimmen, so erreichen Sie alle Zeichen wie im normalen Textmodus.

Sollen die Adressen 53282 oder 53284 die Hintergrundfarbe eines Zeichens bestimmen, so erreichen Sie die Buchstaben durch Drücken der normalen Tasten, die Zahlen und andere gängige Zeichen erhalten Sie durch Drücken von speziellen Tastenkombinationen.

Da es recht mühsam ist, immer die richtige Tastenkombination für ein Zeichen herauszufinden, habe ich Ihnen hier ein Tabelle aufgeführt, an der Sie ablesen können, wie Sie verschiedene Zeichen durch Drücken von Tastenkombinationen auf den Bildschirm bringen können. Hier nun die Tabelle:

Darzustellendes Zeichen	**Tastenkombination**
!	<COMMODORE>-K
"	<COMMODORE>-I
#	<COMMODORE>-T
$	<COMMODORE>-@
%	<COMMODORE>-G
&	<COMMODORE>-+
'	<COMMODORE>-^
(	<COMMODORE>-£
+	<COMMODORE>-Q
−	<COMMODORE>-Z
£	<COMMODORE>-−
*	<COMMODORE>-N
→	<COMMODORE>-*
↑	<COMMODORE>-↑
=	<COMMODORE>-X
;	<COMMODORE>-F

<	<COMMODORE>-C
>	<COMMODORE>-V
?	<COMMODORE>-B
,	<COMMODORE>-D
.	<COMMODORE>-S
/	<COMMODORE>-P
0	<COMMODORE>-A
1	<COMMODORE>-E
2	<COMMODORE>-R
3	<COMMODORE>-W
4	<COMMODORE>-H
5	<COMMODORE>-J
6	<COMMODORE>-L
7	<COMMODORE>-Y
8	<COMMODORE>-U
9	<COMMODORE>-O

Hiermit ist nun alles über den erweiterten Textmodus des C64 gesagt. Experimentieren Sie am besten selbst ein wenig, denn dieser Modus ist sehr leistungsfähig; doch vergessen Sie nicht weiterzulesen!

6.4 Die hochauflösende Grafik

Es gibt viele Programme, die nicht im Textmodus, sondern im Grafikmodus des C64 arbeiten, so z.B. viele Spiele. Der Vorteil in diesem Modus liegt darin, daß jeder einzelne Bildschirmpunkt gesetzt und gelöscht werden kann, wodurch hochauflösende Grafiken möglich sind. Dieser Modus wird nicht von BASIC unterstützt, weshalb er zwar den meisten Benutzern bekannt ist, die wenigsten ihn jedoch programmieren können. Dieses Kapitel soll Ihnen nun zeigen, wie man den hochauflösenden Modus aktiviert und wie man mit ihm arbeitet.

6.4.1 Grundlagen zur Grafikprogrammierung

Wie für alles grafische, ist der VIC auch für den hochauflösenden Modus zuständig. Zuerst muß Ihm mitgeteilt werden, daß der Grafikmodus aktiviert werden soll. Dies geschieht durch setzen des 5.Bits im Register 17 des VIC:

```
POKE 53265,PEEK (53265) OR 32
```

Nun wird das sich Ihnen zeigende Bild nicht gerade die Ordnung in Person sein. Das liegt einfach daran, daß nun ein Speicherbereich für die Darstellung des Bildschirms verantwortlich ist, der zufällige Werte enthält. Da wir auch nicht wissen, welcher Speicherbereich für die Bildschirmdarstellung zuständig ist, geben wir am besten selbst einen an. Dies geschieht durch das Setzen bzw. Löschen des 3. Bits des Registers 24. Um den Speicherbereich von 0-7999 für die Bildschirmdarstellung verantwortlich zu machen, muß das 3. Bit gelöscht werden. Ist es gesetzt, so ist der Bereich von 8192-16191 für die Bildschirmdarstellung verantwortlich.

Da sich der Speicherbereich von 0-7999 nicht so gut für die Speicherung der Bilddaten eignet (Zeropage, BASIC-Speicher), wollen wir den Bereich von 8192 bis 16191 dafür benutzen. Es muß also das 3. Bit der Adresse 53272 gesetzt werden. Das geschieht durch folgenden Befehl:

```
POKE 53272,PEEK (53272) OR 8
```

Nun ändert sich das Aussehen des Bildschirms etwas, Ordnung kommt jedoch immer noch nicht ins Bild. Am besten wird es sein, wenn wir den Bildschirm löschen. Dies geschieht durch POKEn von Nullen in den Bereich, der für die Bilddarstellung zuständig ist, was in diesem Fall der Bereich von 8192 bis 16191 ist. Geben Sie dazu bitte folgende Befehle ein:

```
FOR A=8192 TO 16191 : POKE A,0 : NEXT A
```

Nach Drücken der RETURN-Taste werden Sie sehen, wie der Bildschirm langsam gelöscht wird. Bis jetzt können wir den Grafikmodus einschalten, den Speicherbereich bestimmen, der für die Bilddarstellung verantwortlich ist und den Bildschirm löschen. Was uns jetzt noch fehlt, ist die Bestimmung der Farben, in der die einzelnen Punkte dargestellt werden sollen.

Hier muß ich anmerken, daß der C64 zwei Grafikmodi besitzt. Der eine ist der sogenannte Hi-Res-Modus. In diesem Modus kann jeder Punkt einzeln dargestellt werden. Die Farbe ist je-

doch immer für 64 Punkte (8*8-Punktmatriz) gleich. Im anderen Modus, dem sogenannten Multicolor-Modus ist es möglich, die Farben eines Punktes (fast) beliebig zu bestimmen, was jedoch auch seinen Preis hat: Durch die Farbenvielfalt muß leider die Anzahl der Punkte halbiert werden. Die Auflösung im Multicolor-Modus ist also nur halb so groß wie im Hi-Res-Modus. Aus diesem Grund habe ich mich dafür entschieden, Ihnen hier die Programmierung des Hi-Res-Modus zu zeigen.

Der Hi-Res-Modus ist jedoch gar nicht so farblos, wie es zuerst scheint, denn durch geschickte Programmierung kann man doch beachtliche Ergebnisse erzielen (Die Benutzeroberfläche GEOS arbeitet übrigens auch im Hi-Res-Modus, und so farblos ist GEOS ja schließlich auch nicht!). Als Farbspeicher für den Grafikbildschirm dient das Video-RAM, das sich an den Adressen 1024 bis 2023 befindet. Jede Adresse in diesem Bereich bestimmt die Vorder- und Hintergrundfarbe einer 8*8-Punktmatrix. Um mit einem Byte die Vorder- und Hintergrundfarbe gleichzeitig zu bestimmen, ist eine Unterteilung des Bytes in zwei Nibbles nötig. Ein Nibble beträgt vier Bit. Die unteren Bits (Bit 0-3) geben die Hintergrundfarbe an, während die oberen vier Bits (Bit 4-7) die Vordergrundfarbe angeben. Daraus ergibt sich folgende Formel zur Bestimmung einer Farbe:

```
Farbe = Vordergrundfarbe * 16 + Hintergrundfarbe
```

Diese Werte müssen nun nur noch in die entsprechenden Speicherstellen gepoked werden. Folgende Zeile setzt die Vordergrundfarbe für den gesamten Bildschirm auf weiß (Zahl 1) und die Hintergrundfarbe auf schwarz (Zahl 0):

```
FOR A=1024 TO 2023 : POKE A,1*16+0 : NEXT A
```

Damit haben wir alle wichtigen Grundlagen zur Grafikprogrammierung besprochen. Wir können uns nun daran machen, einzelne Punkte zu setzen und zu löschen. Doch vorher noch einmal eine Zusammenfassung dieses Kapitels. Durch Setzen des 5. Bits in der Speicherstelle 53265 wird der Grafikmodus eingeschaltet, durch Löschen des 5. Bits gelangt man wieder in den Normalmodus. Der Zustand des 3. Bits in der Speicherzelle

53272 gibt an, welcher Speicherbereich für die Darstellung der Grafiken verantwortlich ist. Ist dieses Bit gelöscht, so liegt der Grafikbereich von 0-7999, ist das Bit gesetzt, so liegt er von 8192-16191. Letztendlich wird die Farbe der Punkte durch die Zustände der Adressen 1024-2023 festgelegt. Dabei sind die unteren 4 Bits für die Hintergrundfarbe verantwortlich, während die oberen 4 Bits die Hintergrundfarbe bstimmen.

6.4.2 Setzen von Punkten

Unser nächstes Ziel ist es nun, einzelne Punkte gezielt zu setzen oder zu löschen. Da nicht jeder Punkt einzeln angesprochen werden kann, ist es nicht ohne weiteres möglich, einzelne Punkte anzusprechen. Doch warum ist dies nicht möglich? Sollte jeder Punkt einzeln angesprochen werden, so müßte für jeden Punkt ein Byte reserviert werden. Wenn Sie nun einmal ausrechnen, wieviel Speicher dann eine Grafikseite verbrauchen würde, nämlich 320*200=64000 Bytes, so werden Sie merken, daß dann der Speicher verbraucht wäre. Es wäre also nicht mehr möglich, Programme oder andere Daten unterzubringen.

Da dies sehr unpraktikabel wäre, hat man das Ablegen der Daten folgendermaßen organisiert: Es werden insgesamt acht Punkte zu einem Byte zusammengefaßt. Damit wird der Grafikspeicher um das Achtfache kleiner. Jedes Bit gibt also den Zustand eines Grafikpunktes wieder.

Nun wäre es einleuchtend, wenn die Bytes nacheinander organisiert werden, also das erste Byte den Zustand der ersten acht Punkte wiedergäbe, das zweite Byte den Zustand für die acht Bytes rechts daneben usw. Dies ist jedoch nicht der Fall. Um die Daten intern einfacher bearbeiten zu können, sind immer acht Bytes zu einem Block zusammengefaßt worden; die ersten acht Bytes geben also den Zustand der ersten 64 (8*8) Grafikpunkte wieder, die zweiten acht Bytes den Zustand der zweiten 64 Grafikpunkte usw. Um dieses noch anschaulicher zu machen, möchte ich Ihnen hier einmal in einer Abbildung zeigen, wie die Grafikdaten abgelegt werden.

8192	8200		8504
8193	8201		8505
8194	...	. . . .	...
8195			
8196			
8197			
8198			
8199			
8512			

Abb. 6 Grafikdaten

Solch eine Abspeicherung der Grafikdaten ist insofern leichter zu verarbeiten, als z.B. bei der Farbgebung auch immer die Farben für einen 8-Byte-Block gleichzeitig festgelegt werden. So werden mühsame Umrechnungen erspart. Ein Punkt wird nun dadurch gesetzt, daß das entsprechende Bit gesetzt wird. Soll ein Punkt gelöscht werden, so muß das entsprechende Bit gelöscht werden. Dies geschieht durch die Verknüpfung mittels OR (setzen) oder AND (löschen). Sollte Ihnen dies nicht mehr so geläufig sein, so sollten Sie noch einmal im Anhang nachsehen.

Nun kommt der schwierige Teil dieses Kapitels: Das Errechnen der Adresse, in die ein Wert geschrieben werden soll. Es ist zuerst sinnvoll, die Anzahl der Zeilen durch acht zu teilen und diesen Wert dann zu runden. Da die Daten immer in 8-Byte-Blöcken abgespeichert werden, ist dies sinnvoll, denn so erhalten wir eine Zahl, die wir nur mit 320 multiplizieren müssen, und schon ist der Block, der überhalb der 8 Zeilen, in dem die zu ändernde Zeile liegt, erledigt. Hier nun der Anfang der Formel (y=Zeile)

```
Y=INT(Y/8)*320
```

Wir müssen uns nun von links dem zu ändernden 8-Byte Block nähern, da die 8-Byte-Blöcke, die links davon liegen, unverändert bleiben sollen. Wir müssen also die Anzahl der Spalten durch acht teilen und diesen Wert ebenfalls runden. Dieser Wert muß dann noch mit 8 multipliziert werden, da die Daten in 8-Byte-Blöcken vorliegen, und zu dem obigen Wert addiert werden:

```
Y=INT(Y/8)*320 + INT(X/8)*8
```

Wir haben jetzt die Anfangsadresse des 8 Byte-Blocks vor uns, in dem sich das zu ändernde Byte befindet. Wir brauchen nun noch die Anzahl der restlichen Zeilen, die Anzahl der gesamten Zeilen minus der Zeilen, die wir schon erledigt haben. Dafür können wir folgende Formel benutzen:

```
RESTZEILEN=Y-(INT(Y/8)*)
```

Diese Zahl muß noch zu der anderen Zahl addiert werden. Daraus ergibt sich folgende Formel für die Adresse des zu ändernden Garfikpunktes:

```
ADR=INT(Y/8)*320 + INT(X/8)*8 + (Y-INT(Y/8)*8)
```

Was uns nun noch fehlt, ist das zu ändernde Bit. Dazu müssen wir die Anzahl der schon erledigten Spalten von der Anzahl Spalten subtrahieren. Dies geschieht durch folgende Formel:

```
BIT=X-(INT(X/8)*8)
```

Nun taucht jedoch noch ein Problem auf: Soll das linke Bit verändert werden, so ist das nach dieser Rechnung das 0. Bit. Dies ist jedoch nicht der Fall, da das 7. Bit links und das 0. Bit rechts liegt. Man muß also die einzelnen Bitwerte spiegeln. Dies macht man folgendermaßen: Man subtrahiert von dem erhaltenen Wert 7 und nimmt davon den Absolutwert. Dadurch wird aus 0 eine 7 (0-7=-7 ==> ABS(-7)=7), aus der 3 wird eine 4 (4-7=-3 ==> ABS(-3)=3) usw. Um nun den neuen Wert zu erhalten, muß die Zahl 2 mit diesem Wert potenziert werden (siehe Anhang). Daraus folgt folgende Formel:

```
WERT=2^(ABS(X-(INT(X/8)*8)-7))
```

Am besten machen Sie sich das Ganze einmal durch das Einsetzen bestimmter Werte klar. Suchen Sie sich einen Punkt aus, und rechnen Sie die Formeln durch. Ich habe ein Programm geschrieben, das Ihnen zeigen soll, wie man eine Grafik im hochauflösenden Modus programmiert. Ich habe hier die Darstellung einer Sinuskurve gewählt, da so auch deutlich wird, wie man Formeln grafisch umsetzt.

```
100 POKE 53265,PEEK (53265) OR 32
110 POKE 53272,PEEK (53272) OR 8
120 DIM A$(10000) : CLR
130 FOR A=1024 TO 2023 : POKE A,16 : NEXT A
140 FOR X=0 TO 319
150 Y=INT(SIN(X*2*(PI/180))*90)+100
160 Y=-1*Y+200
170 ADR=INT(Y/8)*320 + INT(X/8)*8 + (Y-INT(Y/8)*8)
180 WERT=2^(ABS(X-(INT(X/8)*8)-7))
190 POKE 8192+ADR,PEEK (8192+ADR) OR WERT
200 NEXT X
210 WAIT 198,1
220 POKE 53265,PEEK (53265) AND 255-32
230 POKE 53272,PEEK (53272) AND 255-8
```

Hier nun noch einige Erläuterungen zum Programm: Die Zeilen 100 und 110 dürften Ihnen bereits bekannt vorkommen. Hier wird lediglich der Grafikmodus eingeschaltet und der Bereich für die Grafikdaten festgelegt. In Zeile 120 kommt nun ein neuer Trick. Hier wird der Grafikbereich einmal auf eine andere (schnellere) Weise gelöscht. Es wird ein Feld von 10000 Strings definiert, dessen Inhalt dann durch den CLR-Befehl gelöscht wird. Da dieses Feld genau auf dem Bereich ab 8192 liegt, wird dieser Bereich auf Null gesetzt. Zeile 130 und 140 sollten Ihnen auch bekannt sein. In Zeile 150 wird nun der Y-Wert zu jedem X-Wert errechnet. Der X-Wert muß vorher noch ins Bogenmaß umgerechnet werden und mit zwei multipliziert werden, da sonst die Sinus-Kurve nicht ganz auf den Bildschirm passen würde. Der Wert, den wir nun erhalten, wird anschließend mit 90 multipliziert, denn ansonsten würden wir nur einen Strich auf dem Bildschirm sehen (Sinuswerte liegen ja bekanntlich 'nur'

zwischen 0 und 1.) Anschließend wird noch die Zahl 100 addiert, damit der Nullpunkt genau in der Mitte des Bildschirms liegt.

Ein Unterschied zwischen dem mathematischen Koordinatensystem und dem Koordinatensystem des C64 liegt darin, daß der C64 den Nullpunkt in der linken oberen Ecke hat, während der Nullpunkt im mathematischen Koordinatensystem links unten liegt. Um diesen 'Fehler' zu beheben, muß der erhaltene Y-Wert mit -1 multipliziert werden, und zu diesem neuen Wert muß noch die Zahl 200 addiert werden (aus dem Y-Wert 10 ergibt sich somit der Wert 190, aus 50 ergibt sich 150 usw.) Dies passiert in Zeile 160.

In Zeile 170 wird die Adresse des zu verändernden Bytes und in Zeile 180 wird der neue Wert für diese Adresse errechnet. Die restlichen Zeilen sind dazu da, um nach Drücken einer Taste (Abfrage in Zeile 210) die Normalwerte wiederherzustellen. Ich habe Ihnen in diesem Kapitel nun eine kleine Einführung in die Grafikprogrammierung gegeben. Sollten Sie Feuer gefangen haben, so kann ich Sie hier auf das ebenfalls im Data Becker Verlag erschienene Supergrafikbuch verweisen.

6.5 Mehrfarbiger Bildschirmrand

Wußten Sie schon, daß man den Bildschirmrand auch bunter als normal gestalten kann? Nein, dann lesen Sie am besten weiter!

6.5.1 Zweifarbiger Bildschirmrand

Geben Sie bitte einmal folgendes Programm ein:

```
10 POKE 53280,10
20 FOR A=0 TO. ::::NEXT
30 POKE 53280,11
40 ::GOTO 10
```

Wenn Sie das Programm starten, wird der Bildschirmrand aus zwei Farben - nämlich aus hellrot und dunkelgrau - bestehen. Der Unterschied dieser Routine zu den 'normalen' Routinen, die die Hintergrundfarbe ändern, besteht darin, daß eine Grenze zwischen den beiden Hintergrundfarben besteht.

Es gibt einen sogenannten Rasterstrahl. Dies ist ein Strahl, der 50mal in der Sekunde den Bildschirm hinunterwandert. Wird nun die Farbe gewechselt, so wird sie ab der Stelle gewechselt, an der sich der Rasterstrahl gerade befindet. Paßt man nun die Zeiten zwischen dem Wechseln der Farben so ab, daß sich der Rasterstrahl immer an derselben Stelle befindet, so erscheint es, als hätte man einen zweifarbigen Bildschirmrand , da die Grenze des Farbwechsels immer gleich ist. Deshalb ist es auch sehr wichtig, daß Sie dieses und die gleich folgenden Programme haargenau so abtippen, wie sie hier abgebildet sind.

Die Doppelpunkte, Leerzeichen etc. bewirken nähmlich eine Verzögerung, damit der Rasterstrahl sich immer an der gewünschten Stelle befindet, sobald die Hintergrundfarbe gewechselt wird. Solch eine Routine ist dann sinnvoll, wenn das Programm z.B. auf das Drücken einer Taste oder ähnliches wartet. Hier nun eine weitere Routine, die vier verschiedene Hintergrundfarben darstellt:

```
10 A=53280
20 :  :
30 FOR B=0 TO.:::::NEXT:POKE A,3:POKE A,4:POKE A,5:POKE A,6:GET
A$:IF A$=""THEN 30
```

Wenn Sie diese Routine starten, wird Ihnen ein langsam scrollender Hintergrund präsentiert, der aus vier Farben besteht. Drücken Sie eine Taste, so entsteht wieder ein einfarbiger Hintergrund. Diese Routine funktioniert nach dem gleichen *Prinzip* wie die erste Routine, jedoch mit vier Farben. Versuchen Sie auch ruhig einmal, die Programme leicht abzuändern. Dadurch wird manchmal das Scrolling schneller, manchmal auch langsamer, oder die Richtung ändert sich. Allgemein gilt: Ein Leerzeichen verlangsamt das Programm wenig, während die Abar-

beitung eines Doppelpunktes schon etwas mehr Zeit in Anspruch nimmt. Hier nun ein Beispiel, wodurch das Scrolling etwas stärker wird:

```
10 A=53280
20 FOR A=0 TO.::::NEXT:POKE A,3:POKE A,4:POKE A,5:POKE A,6:GET A$:
IF A$=""THEN20
```

So kann die Größe der einzelnen Farbbalken geändert werden. Hierzu auch ein Beispiel:

```
10 A=53280
20 POKE A,3:FOR A=0 TO.::::NEXT:POKE A,4:POKE A,5:POKE
A,6:GETA$:IF A$="" THEN 20
```

Auch hier ist es am besten, wenn Sie einfach ein bißchen experimentieren, denn diese Routinen sind nicht endgültig. Ich möchte Ihnen damit nur zeigen, wie man sich solche Kombinationen zusammenstellen kann. Hier nun noch einige weitere Routinen:

```
10 A=53280
20 POKE A,3:FOR A=0 TO 3:::::::::NEXT:POKE A,4:POKE A,5:POKE A,6
30 POKE A,7:POKE A,8:POKE A,9:POKE A,10:GETA$:IF A$="" THEN 20
```

Hier nun eine Routine, die einen sechsfarbigen Rand erzeugt:

```
10 A=53280
20 POKE A,3:POKE A,4:POKE A,5:POKE A,6:POKE A,7 :POKE
A,8:::::::::::GOTO20
```

Hier nun noch weitere vierfarbige Routinen:

```
10 A=53280
20 GET A$:IFA$=""THENPOKE A,10:POKE A,11:POKE A,12: POKE
A,13::::::GOTO 20
```

```
10 A=53280
20 POKE A,10: :GETA$:IF A$=""THEN POKE A,11:POKE A,12:POKE A,13::
POKE A,14:GOTO 20
```

```
10 A=53280
20 FOR A=0 TO 0 STEP0:POKE A,10:GETA$:IFA$="" THEN POKE A,11:FOR
B=1 TO 4 :NEXT:POKE A,12:NEXT
```

6.5.2 Kombination von Hintergrund- und Rahmenfarbe

Bis jetzt haben wir lediglich die Rahmenfarbe geändert. Nun wollen wir die Hintergrundfarbe gleichzeitig mit der Rahmenfarbe verändern. Hier nun die erste Routine:

```
10 A=53280
20 B=53281
30 POKE A,6:POKE B,6:POKE A,7:POKE B,7:POKE A,2:POKE B,2:GOTO30
```

Nachdem Sie diese Routine gestartet haben, scrollt ein Balken von unten nach oben über den Bildschirm. Besonders interessant ist es zu beobachten, daß der Bildschirminhalt nicht verändert wird. Hier nun zwei weitere Farbroutinen:

```
10 A=53280
20 B=53281
30 POKE A,6:POKE B,6:POKE A,7:POKE B,7:POKE A,2:POKE B,2
40 POKE A,5:POKE B,5:GOTO30
```

```
10 A=53280
20 B=53281
30 POKE A,6:POKE B,6:POKE A,7:POKE B,7:POKE A,2:POKE B,2
40 POKE A,12:POKE B,12:POKE A,15:POKE B,15:POKE A,0:POKE B,0:
50 GETA$:IF A$=""THEN30
```

Ich habe Ihnen nun gezeigt, wie Sie durch recht einfache Mittel Ihre Programme "aufmöbeln" können. Die oben gezeigten Routinen sind jedoch nur Anregungen, die Ihnen die Funktionsweise solcher Routinen vedeutlichen sollen. Sie können (oder sollten sogar!) ruhig mit den Programmen experimentieren. Probieren Sie also ruhig einmal aus, was passiert, wenn Sie die Farben ändern, welche Effekte sich ergeben, wenn Sie die FOR-NEXT-Schleifen verändern, mehr Doppelpunkte ins Programm reinbringen und auch mehr Farben benutzen. Viel Spaß beim Experimentieren!

6.6 Verändern des Zeichensatzes

Der Zeichensatz des C64 ist vom BASIC aus leider nicht ohne weiteres änderbar. Deshalb ist es auch den wenigsten bekannt, wie das Verändern des Zeichensatzes vor sich geht. Daß dies jedoch nicht so schwierig ist, soll Ihnen dieses Kapitel zeigen.

Das Aussehen der Zeichen muß ja irgendwo gespeichert sein, es müssen also an einer bestimmten Stelle im ROM die Daten für den Zeichensatz liegen. Für den Zeichensatz ist der ROM-Bereich von 53248-57343 zuständig (4 KByte). Nun leuchtet es ja ein, daß man einfach durch Pokes versucht, diesen Zeichensatz zu ändern. Doch leider liegt der Zeichensatz ja im ROM, das nur gelesen, nicht aber verändert werden kann. Es muß also zuerst einmal dieser Speicherbereich ins RAM kopiert werden, damit man die Daten verändern kann. Dies geschieht durch folgende Routine, die Sie jedoch noch nicht abtippen sollten:

```
10 FOR A=53248 TO 57343
20 POKE A-40960, PEEK (A)
30 NEXT A
```

Die Funktionsweise dieser Routine dürfte eigentlich klar sein: In einer FOR-NEXT-Schleife werden die Werte der Speicherzellen 53248-57343 ausgelesen und in den Bereich von 12288 - 16383 geschrieben. Warum die Daten ausgerechnet in diesen Bereich geschrieben werden, erkläre ich Ihnen gleich.

Des weiteren muß dem Rechner noch mitgeteilt werden, ab welcher Speicherstelle er die Daten für den Zeichensatz findet. Diese Funktion übernimmt die Speicherzelle 53272. Hier gibt das Lo-Nibble, also die unteren vier Bits an, wo der C64 die Daten für den Zeichensatz findet. Hier die Möglichkeiten in einer Tabelle:

Speicherbereich	Wert
0-4095	01
4096-8191	05
8192-12278	09
12288-16383	13

Programmiert man in BASIC, so ist es am sinnvollsten, den Bereich von 12288 - 16383 für die Zeichensatzdaten zu verwenden, da man so noch am meisten Platz für das BASIC-Programm besitzt. Es muß also in der Speicherstelle 53272 der Wert 13 stehen:

```
POKE 53272,PEEK (53272) AND 240 OR 13
```

Diese etwas umständliche Art, die Bits zu setzen, ist leider nötig, da erst die ersten 4 Bits gelöscht werden müssen, bevor das 0., 2. und 3. Bit gesetzt wird. Sonst könnte es nämlich passieren, daß das 1. Bit gesetzt war. Würden nun nur die anderen Bits gesetzt, so wäre das gesamte Lo-Nibble gesetzt, was nicht so sein sollte. Nun ist Ihnen vielleicht auch klar geworden, warum die Werte in die Speicherzellen A-40960 gePOKEd wurden, denn das ist der neue Bereich für den Zeichensatz Des weiteren muß in der Speicherstelle 1 vor dem Kopieren des Zeichsatzes das 3. Bit gelöscht werden. Dadurch wird der Speicherbereich des Zeichensatzes vorübergehend ausgeschaltet. Nach dem Kopieren muß das 3. Bit natürlich wieder gesetzt werden. Hier nun die beiden Befehle, die das Ein- bzw. Ausschalten erledigen:

```
POKE 1,PEEK (1) AND 251   ' ausschalten
POKE 1,PEEK (1) OR 4      ' einschalten
```

Des weiteren muß vor dem Kopieren das Bit 0 der Speicherstelle 56334 gelöscht werden. Dies geschieht durch folgenden Befehl:

```
POKE 56334,PEEK (56334) AND 254
```

Nach dem Kopieren muß auch dieses Bit wieder gesetzt werden, was durch den Befehl

```
POKE 56334,PEEK (56334) OR 1
```

geschieht. Nun wissen wir alles, um den Zeichensatz ins RAM zu kopieren und ihn dann dort zu verändern. Hier nun das gesamte Programm.

```
10 POKE 56334,PEEK (56334) AND 254
20 POKE 1,PEEK (1) AND 251
30 FOR A=53248 TO 57343
40 POKE A-40960,PEEK (A)
50 NEXT A
60 POKE 1,PEEK (1) OR 4
70 POKE 56334,PEEK (56334) OR 1
80 POKE 53272,PEEK (53272) AND 240 OR 13
```

Der Zeichensatz befindet sich nun im RAM ab der Adresse 12288, wo wir ihn nun verändern können. Spätestens jetzt kommt die Frage auf, wie wir den Zeichensatz eigentlich verändern wollen. Hierfür gibt es mehrere Möglichkeiten. Sie können z.B. einen eigenen Zeichensatz für Ihr neues Programm erstellen, eigene Sonderzeichen kreieren oder sogar durch Verändern des Zeichensatzes Grafiken erstellen. Ich möchte Ihnen diese Methoden hier einmal vorstellen:

Als erstes ist es wohl sinnvoll, ein neues Zeichen zu entwickeln. Nehmen Sie sich dazu ein Blatt, malen Sie auf das Blatt ein 8*8-Raster, und entwerfen Sie nun Ihr Zeichen. Soll dabei ein Punkt erscheinen, so malen Sie das entsprechende Kästchen aus, soll kein Punkt erscheinen, so lassen Sie das Kästchen unversehrt. Ich habe folgendes Zeichen entworfen:

```
****....    240
*..*....    144
*..*....    144
*******.    254
*..*..*.    146
*..****.    158
...*....     16
...*....     16
```

Daraus ergeben sich folgende Werte für unser neues Zeichen:

```
240,144,144,254,146,158,16,16
```

Diese Werte müssen nun nur noch in die entsprechenden Speicherstellen gePOKEd werden. Nehmen wir einmal an, Sie wollen das Klammeraffen-Zeichen durch dieses neue Zeichen ersetzen. Geben Sie dazu folgende Zeilen ein:

```
10 FOR A=12288 TO 12288+7
20 READ B
30 POKE A,B
40 NEXT A
50 DATA 240,144,144,254,146,158,16,16
```

Nachdem Sie das Programm gestartet haben, sollte nun beim Drücken des Klammeraffen unser neues Zeichen erscheinen. Warum haben wir die Werte genau in dieses Speicherzellen gePOKEd? Nun, genau an dieser Stelle befinden sich die Daten für das Zeichen des Klammeraffens. Der Klammeraffe hat ja bekanntlich den Character-Code Null. Deshalb liegen auch die Werte, die dieses Zeichen definieren, direkt am Anfang des ganzen Zeichensatzes. Danach kommen die Daten für das Zeichen A, dann B usw. Ab dem Character-Code 65 kommen die Daten für die Grafikzeichen. Ab Character-Code 128 folgen dann die ganzen Zeichen noch einmal in reverser Schrift.

Möchten Sie eine Grafik erstellen, ohne direkt im Grafikmodus zu arbeiten, so geht dies auch recht einfach mittels Veränderung des Zeichensatzes. Hierzu ein kleines Beispiel. Nehmen wir einmal an, Sie wollen ein Haus zeichnen, das ungefähr so aussieht:

```
.......*    1      ........    0
......*.    2      *.......  128
.....*..    4      .*......   64
....*...    8      ..*.....   32
...*....   16      ...*....   16
..*.....   32      ....*...    8
.*......   64      .....*..    4
********  255      *******.  254
*.....**  131      **....*.  194
*.....*.  130      .*....*.   66
*.....**  131      **....*.  194
*.......  128      ......*.    2
*.....**  131      **....*.  194
*.....*.  130      .*....*.   66
*.....*.  130      .*....*.   66
*.....*.  130      .*....*.   66
```

Anstelle der Buchstaben A-D sollen nun folgende Zeichen erscheinen. Dazu müssen die Daten in die entsprechenden Speicherstellen gePOKEd werden. Dies geschieht durch folgendes Programm:

```
10 FOR A=12296 TO 12296+31
20 READ B
30 POKE A,B
40 NEXT A
50 DATA 1,2,4,8,16,32,64,255
60 DATA 0,128,64,32,16,8,4,254
70 DATA 131,130,131,128,131,130,130,130
80 DATA 194,66,194,2,194,66,66,66
```

Drücken Sie nun erst die Taste A, dann B. Fahren Sie nun mit dem Cursor auf die nächste Zeile, und drücken Sie erst die Taste für das eigentliche C und dann die Taste für das D. Nun können Sie Ihr Haus betrachten. Die Voraussetzung für diese Zeichensatzänderungen ist natürlich, daß der Zeichensatz vorher ins RAM kopiert wurde.

6.7 Verändern des Cursors

Ist Ihnen nicht schon einmal aufgefallen, daß der Cursor bei Personal-Computern oder anderen Rechnern viel professioneller aussieht als beim C64? Beim PC besteht der Cursor z.B. nur aus einem Strich. Der Cursor des C64 läßt sich jedoch verändern. Um dies zu verwirklichen, muß ich Ihnen erst einmal ein paar Grundlagen vermitteln, wie der Cursor dargestellt wird.

Der Cursor ist nichts anderes als das Revers-Zeichen des Zeichens, auf dem er sich gerade befindet. Steht der Cursor also z.B. auf der Zahl 1, so wird alle halbe Sekunde dieses Zeichen invertiert. Es erscheint also eine reverse 2. Um noch genauer zu sein: Der C64 stellt das Zeichen dar, das er an der gleichen Stelle in der zweiten Hälfte des Zeichensatzes findet, wie das erste eigentliche Zeichen. Nun dürfte es Ihnen nicht schwerfallen, den Cursor mittels Veränderung des Zeichensatzes so zu verändern, daß er nur als Strich erscheint.

Es muß also erst einmal der Zeichensatz ins RAM kopiert werden. Das Programm dazu finden Sie in Kapitel 6.6. Danach müssen die inversen Zeichen so abgeändert werden, daß sie genauso aussehen wie die normalen Zeichen. Dies geschieht durch folgende Routine:

```
10 FOR A=12288+4096/4 TO 12288+4096/2
20 POKE A,PEEK (A-4096/4)
30 NEXT A
```

Dieses Programm sollten Sie jedoch noch nicht abtippen, da die reversen Zeichen so aussehen wie die normalen. Es ist also kein Cursor sichtbar. Um den Cursor nun in einen Strich zu verwandeln, muß nun das 7. Byte den Wert 255 erhalten, wodurch immer nur die unterste Reihe der Zeichen invertiert wird. Geben Sie dazu bitte folgendes Programm ein:

```
40 FOR A=12288+4096/4+7 TO 12288+4096/2 STEP 8
50 POKE A,255
60 NEXT A
```

Sollten Sie das Programm nun starten, so müssen Sie etwas warten; doch das Warten lohnt sich, denn danach haben Sie einen professionellen Cursor, der nur aus einem Strich besteht. Haben Sie nun einmal zufällig in den Kleinschrift-Modus umgeschaltet, so werden Sie gemerkt haben, daß hier der Cursor noch normal ist. Das liegt daran, daß wir bis jetzt erst die erste Hälfte des Zeichensatzes geändert haben. Die erste Hälfte des Zeichensatzes ist nämlich nur für den Großschrift-Modus zuständig. Schaltet man in den Kleinschrift-Modus, so gibt die obere Hälfte des Zeichensatzes das Aussehen der Zeichen an.

Wollen wir also auch im Kleinschrift-Modus den Cursor nur als Strich haben, so müssen Sie auch die zweite Hälfte des Zeichensatzes entsprechend ändern. Ich habe Ihnen hier nun noch einmal das ganze Programm aufgeführt, das erst den Zeichensatz ins RAM kopiert und dann die reversen Zeichen so abändert, daß der Cursor nur als Strich erscheint:

```
10 POKE 56334,PEEK (56334) AND 254
20 POKE 1,PEEK (1) AND 251
30 FOR A=53248 TO 57343
40 POKE A-40960,PEEK (A)
50 NEXT A
60 POKE 1,PEEK (1) OR 4
70 POKE 56334,PEEK (56334) OR 1
80 POKE 53272,PEEK (53272) AND 240 OR 13
90 FOR A=12288+4096/4 TO 12288+4096/2
100 POKE A,PEEK (A-4096/4)
110 NEXT A
120 FOR A=12288+4096/4+7 TO 12288+4096/2 STEP 8
130 POKE A,255
140 NEXT A
150 FOR A=12288+(3/4)*4096 TO 12288+4096
160 POKE A,PEEK (A-4096/4)
170 NEXT A
180 FOR A=12288+(3/4)*4096+7 TO 12288+4096 STEP 8
190 POKE A,255
200 NEXT A
```

Starten Sie dieses Programm, so erscheint der Cursor im Groß- und Kleinschriftmodus nur noch als Strich. Ein Nachteil ist jedoch nicht zu verleugnen: Dadurch, daß die reversen Zeichen abgeändert wurden, stehen Ihnen im Revers-Mode auch nur noch die abgeänderten Zeichen zur Verfügung. Hier noch eine Routine, die den Cursor als senkrechten Strich erscheinen läßt. Da sich die nun folgende und die obige Routine nur ab Zeile 90 unterscheiden, sollten Sie die Zeilen 10-80 übernehmen. Die Routine sieht folgendermaßen aus:

```
90 FOR A=12288+4096/4 TO 12288+4096/2
100 POKE A,PEEK (A-4096/4)
110 NEXT A
120 FOR A=12288+4096/4 TO 12288+4096/2-1
130 POKE A, PEEK (A) OR 128
140 NEXT A
150 FOR A=12288+(3/4)*4096 TO 12288+4096
160 POKE A,PEEK (A-4096/4)
170 NEXT A
180 FOR A=12288+(3/4)*4096 TO 12288+4096
190 POKE A,PEEK (A) OR 128
200 NEXT A
```

6.8 Invertieren einer Grafik

Manchmal kann es recht nützlich sein, einen Grafikbildschirm zu invertieren. Dies ist z.B. recht nützlich, wenn die Bildschirmfarben so nicht mehr gut zu erkennen sind. Doch wie immer, so vermißt man auch hier den Befehl, der dieses tut. Deshalb habe ich Ihnen eine kleine Routine geschrieben, die dieses für Sie in atemberaubender Schnelligkeit erledigt:

```
10 FOR A=8192 TO 16191
20 B=PEEK (A)
30 POKE A,(255-B) AND (255-B)
40 NEXT A
```

Diese Routine sieht zuerst etwas verwirrend aus, weshalb ich sie Ihnen kurz erläutern möchte. Es wird zuerst eine Schleife aufgebaut, die genau 7999 mal durchlaufen wird. Dabei steht der Schleifenwert für die Adresse des Grafikbildschirms, die invertiert werden soll. Der Grafikbildschirm muß also bei Adresse 8192 beginnen. In Zeile 20 wird der Variablen B der Wert der Speicherstelle A zugewiesen. Das eigentliche Invertieren folgt nun in Zeile 30. Die Formel

```
(255-B) and (255-B)
```

bewirkt, daß erst die Bits gesetzt werden, die vorher nicht gesetzt waren. Es sind also alle Bits dieses Bytes positiv. Anschließend werden die Bits gelöscht, die vorher gesetzt waren. Dies klingt alles recht kompliziert; deshalb hier einmal ein Zahlenbeispiel. Angenommen, B hat den Wert 10, dann wird folgendes gerechnet:

```
(255-10) AND (255-10)
```

Es werden also erst alle Bits außer dem 1. und 3. gesetzt (die jedoch vorher schon gesetzt waren), und danach werden das 1. und das 3. Bit gelöscht. Ich muß zugeben, daß diese Routine sehr langsam ist. Es ist natürlich klar, daß Sie nur das Ergebnis dieser Routine bemerken, wenn Sie sich im Grafikmodus befin-

den. Sie können diese Routine z.B. in unser Sinuskurven-Programm einbauen. Laden Sie dazu das Programm, und ergänzen Sie folgende Zeilen:

```
192 FOR A=8192 TO 16191
194 B=PEEK (A)
196 POKE A,(255-B) AND (255-B)
198 NEXT A
```

Starten Sie nun das Programm, so wird erst die Kurve gezeichnet. Danach wird sie dann invertiert.

6.9 Ein kleiner Sprite-Editor

Wenn Sie Sprites programmieren wollen, so müssen Sie erst die Grafikdaten Ihres Sprites in Zahlenwerte umrechnen, um diese dann in DATA-Zeilen ablegen zu können. Dies ist jedoch sehr umständlich. Ich möchte Ihnen an dieser Stelle zwei Methoden zum Entwerfen von Sprites bekanntmachen. Die erste besteht darin, ein Raster von 24*21 Punkten aufzumalen und die Punkte, die gesetzt werden sollen, auszumalen. Danach können Sie die Wert nach der bekannten Methode zusammenrechnen.

Jetzt werden Sie sich fragen, warum das Ganze denn so altmodisch mit Blatt und Papier geschieht. Ich kann Ihnen hier nur Recht geben. Deshalb möchte ich Ihnen hier ein Programm präsentieren, das die DATA-Werte selbst errechnet. Bitte tippen Sie einmal folgendes Programm ab:

```
10 FOR A=0 TO 2
20 WERT=0
30 FOR B=0 TO 7
40 WERT=WERT-2^(7-B)*PEEK (1024+B+A*8)=43)
50 NEXT B
60 PRINT A;
70 NEXT A
80 PRINT
90 C=C+1
100 IF C<21 THEN 10
```

Starten Sie das Programm jedoch noch nicht. Es funktioniert folgendermaßen: Löschen Sie den Bildschirm. 'Malen' Sie nun an die Stelle ein +-Zeichen, an denen in Ihrem Sprite ein Punkt gesetzt werden soll. Sie können dabei beliebig mit dem Cursor auf dem Bildschirm umherfahren. Passen Sie jedoch auf, daß Sie Ihr Sprite nicht durch Drücken der <RETURN>-Taste oder durch Scrollen zerstören. Das Eingabefeld hat eine Größe von 24 Spalten mal 21 Zeilen. Haben Sie nun Ihr Sprite entworfen, so fahren Sie bitte mit dem Cursor auf die vorletzte Zeile und geben Sie

```
RUN
```

ein. Es werden nun nach und nach die Daten für Ihr Sprite ausgegeben. Die Funktionsweise ist eigentlich recht einfach: Es werden zwei Schleifen durchlaufen, in denen überprüft wird, ob ein +-Zeichen in einer Speicherstelle steht. Ist dies der Fall, so wird der entsprechende Bit-Wert zu dem Variablenwert addiert. Daraus ergeben sich dann die Werte für das Sprite.

Haben Sie die Erläuterung zur Funktionsweise dieses Programmes nicht ganz verstanden, so ist dies auch nicht so schlimm. Am besten gucken Sie sich das Programm noch einmal in aller Ruhe an und überlegen, wie Bildschirmdaten abgespeichert werden.

7. Tips und Tricks zur Soundverbesserung

Daß der C64 recht gute Soundmöglichkeiten besitzt, weiß mittlerweile wohl jeder C64-Besitzer. Doch wie kann man diese nutzen? Ich kann Ihnen hier (leider) nicht eine Einführung in die Soundprogrammierung auf dem C64 bieten, denn dazu ist dieses Buch nicht da. Ich muß an dieser Stelle auf weitergehende Literatur verweisen, z.B. auf das im DATA BECKER Verlag erschienene Musikbuch zum C64. Dieses Kapitel soll Ihnen ein paar Tips und Anregungen bieten, wie Sie die Soundprogrammierung auf dem C64 optimieren können. Bekanntlich reichen hier ja oft einige POKEs aus.

7.1 Sound aus der Datasette

Die Datasette arbeitet genauso wie ein ganz normaler Kassettenrecorder. Deshalb sollte es doch möglich sein, ihr auch einmal den neusten Discosound oder Beethovens Neunte zu entlocken. Daß dieses möglich ist, zeigt - wie so oft - dieser Poke:

```
POKE 54296,15
```

Laden Sie nun ein Programm von Datasette, so werden Sie nun ein Gepiepse hören, was nichts anderes als die Daten sind, die momentan in den Rechner geladen werden. Das funktioniert recht einfach: Genauso wie beim Fernseher, kann man auch beim SID (dies ist der Chip, der für den Sound zuständig ist) die Lautstärke regulieren. Dies geschieht im Register Nummer 25 des SID. Dabei steht der Wert 15 für die höchste und der Wert 0 für die niedrigste Lautstärke. Oben genannter Poke stellt also die Lautstärke auf maximal ein, wodurch die Töne der Datasette zu hören sind.

Es ist natürlich auch möglich, hier kleinere Werte einzugeben, was zur Folge hat, daß die Töne immer leiser werden. Sie sollten

allerdings von der musikalischen Qualität der Datasette nicht so viel erwarten, da sie nicht dafür gedacht ist, Musik wiederzugeben.

7.2 Der Soundfilter

Leider rauscht der C64 ab und zu, wenn er irgendwelche Geräusche von sich gibt. Daß sich dieses Manko auch unterbinden läßt, soll dieser Trick zeigen. Bitte geben Sie in Ihr Sound-Programm folgenden Poke ein:

```
POKE 54295,PEEK (54295) OR 4
```

Das Rauschen des C64 sollte nun weitgehend abgeschaltet sein. Wie funktioniert dies? Nun, der SID besitzt den sogenannten Resonanzfilter. Wenn man in dieser Speicherstelle das 4. Bit setzt, so wird das Rauschen weitgehendst unterbunden. Es werden nur noch klare und saubere Töne gespielt.

7.3 Verbesserung des Sounds

Der Sound läßt sich etwas verbessern. Vielleicht ist Ihnen schon einmal aufgefallen, daß der Rhythmus oft nicht so ganz stimmt, obwohl Ihre eingegebenen Töne die richtige Länge haben. Sie können aber beruhigt sein, die musikalischen Fähigkeiten des C64 sind nicht eingeschränkt, denn dies ist eine ganz normale Erscheinung. Während der C64 Musik spielt, muß der Bildschirm immer wieder aufgebaut werden. Da dies aber Zeit in Anspruch nimmt, kann es schon mal zu Rhythmusstörungen kommen.

Eine Möglichkeit, solche Rhythmusstörungen zu unterbinden, besteht darin, einfach den Bildschirm auszuschalten; softwaremäßig natürlich. Dies geschieht durch folgenden Poke, der auch im Direktmodus seinen Dienst tut:

```
POKE 53265,11
```

Haben Sie diesen Poke im Direktmodus eingegeben, so ist der Bildschirm jetzt abgeschaltet. Sie müssen nun alle Eingaben blind eintippen. Starten Sie nun ein Soundprogramm, so erhält man einen richtigen Rhythmus. Wollen Sie jedoch auch mal wieder etwas sehen, so können Sie den Bildschirm wieder durch

```
POKE 53265,27
```

einschalten. Sie werden nun auch Ihre Eingaben wieder sehen. Das Abschalten des Bildschirms bewirkt übrigens nicht nur, daß mehr Rhythmus in Ihre Sounds kommt. Die gesamte Arbeitsgeschwindigkeit wird dadurch ein bißchen erhöht (siehe Kapitel über Floppy).

7.4 Abstellen eines Tones

Eigentlich sollte ein Ton, nachdem er beendet wurde, auch wirklich nicht mehr ertönen. Daß dies nicht so ist, hat die Realität schon oft gezeigt. Meistens klingt der Ton dann noch leise im Hintergrund mit. Dies läßt sich einfach dadurch umgeben, daß Sie die Lautstärke des SID so regulieren, daß sie nach der Beendigung eines Tones ganz gering ist. Soll dann wieder ein Ton gespielt werden, so wird wieder die volle Lautstärke eingestellt. Um dies zu realisieren, ist uns wieder das Register 24 des SID's behilflich. Wie schon in Kapitel 7.1 erwähnt, wird die Lautstärke durch

```
POKE 54296,0
```

ausgeschaltet. Sie sollten diesen Poke nach jedem gespielten Ton eingeben, damit dieser nicht nachhallt. Wollen Sie dann wieder Töne spielen, so sollten Sie vorher durch

```
POKE 54296,15
```

die Lautstärke wieder auf maximal stellen. Die oben genannte Methode stellt den Ton jedoch nicht wirklich aus, sondern es wird nur die Lautstärke auf minimal gesetzt. Wollen Sie jedoch einmal einen Ton 'richtig' ausstellen, so müssen Sie die Register

4, 11 bzw. 18 auf Null setzen. Jedes Register ist dafür zuständig, ob ein Kanal (!) an- oder ausgestellt ist. Wollen Sie z.B. den Kanal 1 ausstellen, so geschieht dies durch

```
POKE 54276,0
```

Wollen Sie alle drei Kanäle ausstellen, so geschieht dies durch:

```
POKE 54276,0
POKE 54283,0
POKE 53290,0
```

8. Sonstiges

8.1 Sperren und Steuern der Groß-Klein-Umschaltung

Da hat man eine tolle Bildschirmgrafik mit den Grafikzeichen entworfen, und nun kommt der böse Anwender daher und stellt einen durch Drücken von <Shift-C> bloß. Aber Sie wissen das natürlich zu verhindern. Nach einem

```
POKE 657,128
```

oder

```
PRINT CHR$(8)
```

ist nämlich diese Tastenkombination gesperrt, sooft es der Anwender auch versuchen mag. Das Tolle daran: Auch weiterhin sind alle anderen Tastenkombinationen benutzbar, so daß ein ganz normales Programm ablaufen kann, nur mit dem Unterschied, daß man nun nicht mehr in den Textmodus umschalten kann. Zumindestens Ihr Anwender kann dies nun nicht mehr, Sie jedoch haben weiterhin die volle Souveränität über den Bildschirmmodus. Mit

```
POKE 53272,21
```

wird der Groß-Kleinschrift-Modus aktiviert, und

```
POKE 53272,23
```

aktiviert den Großschrift-Grafikmodus. Mit

```
POKE 657,0
```

oder

```
PRINT CHR$(9)
```

hat dann der ganze Spuk ein Ende, und Sie können nach Herzenslust wieder mit den Tasten C=/Shift in den Textmodus umschalten.

8.2 Mini-Textverarbeitung

Sie haben noch keine Textverarbeitung, müssen aber dringend einen Brief schreiben? Kein Problem, hier kommt die kürzeste uns bekannte zum C64 erhältliche Textverarbeitung: Geben Sie Ihren Text ganz normal wie ein BASIC-Programm ein:

```
10 SEHR GEEHRTES DATA BECKER !
20 IHR BUCH IST VÖLLIGER UNFUG
30 SIE WISSEN SCHON, WARUM.
40 MIT FREUNDLICHEN GRÜßEN,
50 ICH
```

Durch

```
POKE 22,35
```

wird die List-Routine so modifiziert, daß keine Zeilennummern und keine Ready-Meldung beim Listen erscheint, Sie also wie von einer Textverarbeitung gewöhnt den Text auf dem Bildschirm dargestellt bekommen. Mehr noch: Sogar eine Druckfunktion beinhaltet unsere Textverarbeitung: Mit folgendem Befehl wird unser wichtiges Dokument auf dem Drucker ausgegeben:

```
OPEN 1,4:CMD 1:LIST:CLOSE 1
```

8.3 Re-NEW

Kommen wir zu einem sehr wichtigen Thema. Stellen Sie sich vor, Sie haben ein Programm versehentlich mit NEW gelöscht. Oder, was fast noch häufiger vorkommt, Ihnen ist der Rechner abgestürzt, und Sie mußten einen Reset auslösen (hoffentlich verfügen Sie über einen Reset-Taster). Nun gilt wohl als erstes

der altbekannte Spruch, hier durchaus konstruktiv gemeint: Ruhe ist die erste Bürgerpflicht.

Sofern Sie seit dem vermeintlichen Löschen des Programmes keine neuen Zeilen eingegeben oder Variablen angelegt haben, ist das Programm nämlich gar nicht gelöscht, sondern es wurde nur der Zeiger für das BASIC-Ende auf den BASIC-Anfang gelegt, was man leicht ändern kann. Nach Eingeben der folgenden Befehle im Direktmodus ist die Welt wieder in Ordnung:

```
POKE 2050,0:SYS42291:POKE 46,PEEK(35)-(PEEK(781)>253):
POKE 45,PEEK(781)+2 AND 255
```

Diese Befehlsfolge paßt übrigens so gerade in eine 80-Zeichen-Doppelzeile. Bitte lassen Sie sich nicht von der Kürze der anderen Version (Poke 46,peek(35)) verführen, hier wird das BASIC-Ende nicht genau an das Programmende gesetzt, was man natürlich zunächst nicht merkt.

8.4 "Formula too Complex"-Error

Glücklicherweise kommt dieser Fehler auf dem C64 recht selten vor, normalerweise wird von dem Interpreter geduldig fast alles geschluckt, was man ihm so an Formeln eingibt. Wenn nun dieser Fehler aber auftritt, dann richtig. Offensichtlich haben die Programmierer des Betriebssystems vergessen, nach der Ausgabe der Fehlermeldung die Zeiger korrekt zurückzusetzen. Mit einem einfachen Poke geht es dann aber korrekt weiter. Die BASIC-Welt ist wieder in Ordnung.

```
POKE 24,0
```

8.5 Nachladen von Maschinenprogrammen

Vielleicht haben Sie schon einmal aus einer Zeitschrift eine kleine Maschinenroutine abgetippt und dann in der Beschreibung etwa die folgenden Worte gelesen: Laden Sie das Programm mit

```
LOAD "Programmname",8,1
```

vor Ihrem BASIC-Programm, geben Sie dann

```
NEW
```

ein, und laden Sie erst jetzt Ihr BASIC-Programm. Der Sinn liegt darin, daß es im Betriebssystem des C64 zwar vorgesehen ist, Programme absolut zu laden, nach dem Laden wird aber trotzdem der Zeiger für den Variablenanfang, wie es bei BASIC-Programmen durchaus sinnvoll ist, auf die Adresse des zuletzt eingelesenen Bytes gesetzt. Dies ist natürlich bei Maschinenprogrammen ganz und gar nicht sinnvoll, da nun auf einmal der Variablenanfang z.B. irgendwo bei der Adresse 52000 liegen kann, was einem der BASIC-Interpreter unweigerlich als "Out of memory" auslegen wird.

Nun setzt zwar der NEW-Befehl den Variablenanfang korrekt zurück, doch hat dieser den entscheidenden Nachteil, daß man ihn aus naheliegenden Gründen nicht während eines BASIC-Programmes anwenden kann. Die folgende Methode arbeitet demnach auch nicht mit dem New-Befehl, sondern mit dem CLR-Befehl, der nur die Variablen, nicht aber das Programm löscht:

```
POKE 251,PEEK(45)
POKE 252,PEEK(46)
LOAD "PROGRAMMNAME",8,1
POKE 45,PEEK(251)
POKE 46,PEEK(252)
CLR
```

Wenn diese Befehlsfolge am Anfang Ihres Programmes steht, dürfte es nicht weiter stören, daß die Variablen gelöscht werden.

8.6 Cursor beschleunigen

Wenn Ihnen der Cursor zu langsam blinkt, Sie so schnell tippen können, daß der Interrupt (60mal in der Sekunde) nicht mehr alle Ihre Tasten erfaßt, Ihnen die Systemuhr TI$ zu langsam

geht oder Sie sonst eine mehr oder weniger sinnvolle Anwendung für diesen Befehl wissen, können Sie mit

```
POKE 56325,X
```

die Geschwindigkeit des Interrupts erhöhen. X sollte dabei aber nicht kleiner als 10 werden, da für die Abarbeitung des Interrupts auch Zeit benötigt wird und dieser nicht wieder auftreten darf, bevor der alte abgearbeitet ist.

8.7 Ermitteln der File-Parameter

Stellen Sie sich vor, Sie haben ein Programm geschrieben, das Daten von Kassette oder Diskette nachladen muß, und wollen dieses auf verschiedenen Geräten laufen lassen. Nun wissen Sie aber gar nicht im voraus, welches der beiden Geräte angeschlossen ist. Anstatt den Anwender danach zu fragen, gibt es eine wesentlich elegantere Lösung. Mit den folgenden Peeks kann man die File-Parameter des letzten Umgangs mit Peripheriegeräten feststellen:

```
LF=PEEK(184)  logische Filenummer
SE=PEEK(185)  Sekundäradresse
GA=PEEK(186)  Gerätenummer
```

Fragen Sie einfach zu Beginn Ihres Programmes die Gerätenummer ab, und öffnen Sie Ihr Datenfile dementsprechend auf der Floppy oder der Datasette. Bitte beachten Sie, daß auch alle anderen Geräte hier verzeichnet sind. Wenn Sie z.B. vorher eine Ausgabe auf den Drucker gemacht haben, läßt sich nicht mehr feststellen, woher das Programm geladen wurde, da die Geräteadresse in jedem Fall 4 sein wird.

8.8 Lottozahlen mit dem C64

Im Gegensatz zu den sonstigen Spielereien dieses Buches hier endlich mal eine ernsthafte und absolut seriöse Anwendung. Mit dem folgenden Programm lassen sich die Lottozahlen der näch-

sten Woche mindestens ebenso sicher vorhersagen wie mit Tarotkarten, Kaffeesatz oder ähnlichem:

```
10 INPUT "WANN UND WO SIND SIE GEBOREN ";G$
20 FOR T=1 TO 6
30 : Z%(T)=RND(I)*49+1
40 : FOR I=1 TO I-1:IF Z%(I)=Z%(T) THEN I=9:NEXT I
50 : IF I>8 THEN 20
60 NEXT T
70 PRINT "HIER KOMMEN IHRE GANZ PERSÖNLICHEN LOTTOZAHLEN:"
80 FOR T=1 TO 6:PRINT Z%(T);:NEXT T
```

8.9 Datumsberechnung

Oft verlangt man während eines Programmes Datumseingaben. Im folgenden möchte ich Ihnen hierfür zwei kurze Routinen vorstellen. Die erste Routine überprüft, ob ein Datum überhaupt möglich ist. Einen 31. Februar wird es dann künftig nicht mehr geben. Es wird bei beiden Routinen davon ausgegangen, daß das Datum bereits eingelesen wurde und in den Variabelen TA=Tag, MO=Monat und JA=Jahr gespeichert wurde. Hier die Plausibilitätsprüfung für das Datum:

```
10 IF MO<0 OR MO>12 OR TA<1 OR TA>31 THEN 100
20 IF (MO=4 OR MO=6 OR MO=9 OR MO=11) AND TA>30 THEN 100
30 IF MO<>2 THEN 60
40 IF TA>29 THEN 100
50 IF TA=29 AND JA/4<>INT(JA/4) THEN 100
60 PRINT "DATUM MÖGLICH"
70 END
100 PRINT "DATUM UNMÖGLICH"
```

Nicht beachtet wurde die Regel, daß der Schalttag dann ausfällt, wenn das Jahr ein volles Jahrhundert ist und das Jahrhundert nicht durch 4 teilbar ist. Dies war das letzte Mal im Jahr 1900 der Fall, und wird das nächste Mal im Jahr 2100 der Fall sein, ist also für die Praxis unerheblich.

Wochentagsberechnung

```
10 M=MO-2:JH=INT(JA/100):JE=JA-JH
20 IF M<=0 THEN M=M+12:JE=JE-1:
30 IF JE<0 THEN JE=99:JH=JH-1
40 WT=TA+INT(2.6*M-0.2)+INT(JE/4+JE)+INT(JH/4)-2*JH
50 WT=WT-INT(WT/7)*7-(WT<0)*7
```

Nach Durchlauf durch diese Routine steht in der Variablen wt eine Zahl für den Wochentag, 0=Sonntag, 1=Montag .. 6=Samstag.

8.10 Geheimcode

Haben Sie geheime Daten, die niemand zu Gesicht bekommen soll? Oder Sie wollen Ihre Programme so kodieren, daß man ohne Paßwort nicht an sie herankommt? Wie dem auch sei, hier folgt der sicherste mir bekannte Codieralgorithmus: Der Algorithmus arbeitet vom Prinzip her mit einem Paßwort, doch wird dieses nicht, wie gewohnt, als Zugangskontrolle verwendet, sondern als Schlüssel zu einer Funktion, mit der jedes einzelne Byte verschlüsselt wird. Die Grundlage dieser Funktion ist die sog. Exclusive-Or-Verknüpfung, ein Operator, der im C64-BASIC leider fehlt, aber dennoch als kurze Unterroutine programmiert werden kann. Hier die Wahrheitstabelle der EOR-Verknüpfung:

	0	1
0	1	1
1	1	0

EOR arbeitet also genauso wie ein normales OR, nur wenn beide Bits 1 sind, ist das Ergebnis 0. Das Besondere an dieser Verknüpfung ist, daß sie umkehrbar ist:

```
(a EOR b) EOR b = a
```

Man kann daher Byte für Byte alle Daten mit einem nur Ihnen bekannten Schlüsselwert EOR-verknüpfen, und das Ergebnis, das nicht mehr zu identifizieren ist, abspeichern. Zum Dekodie-

ren EOR-verknüpft man einfach die Daten erneut mit dem alten Schlüssel. Dies kann natürlich nur von solchen Personen getan werden, die den Schlüssel kennen. Einen Haken hat die Sache allerdings. In deutschsprachigen Texten haben die einzelnen Buchstaben eine erstaunlich konstante durchschnittliche Häufigkeit, wie aus der folgenden Tabelle zu entnehmen ist:

Buchstabe	Relative Häufigkeit in %
SPACE	14.4
E	14.4
N	8.7
S	6.5
I	6.3
R	6.2
A	5.9
D	5.5
T	5.4
U	4.2

Werden nun zwei verschiedene Bytes gefunden, die in dem unbekannten Text etwa 15% aller Bytes stellen, und ein drittes, welches etwa 9% aller Bytes stellt, so ist klar, daß dieses 3. ein N war. Durch Rückrechnen kommt man dann sehr schnell auf den Schlüssel und kann auch alle anderen Zeichen identifizieren. Mit diesem Kunstgriff können übrigens alle Codes entschlüsselt werden, die eindeutig jedem Ausgangszeichen ein anderes Zeichen zuordnen, egal, wie kompliziert die Methode hierzu auch sein mag. Die Methode ist also erweiterungsbedürftig.

Normalerweise benutzt man daher den eingegebenen Wert nur als Startwert, die folgenden Schlüssel werden nach einem bestimmten Algorithmus aus dem jeweils vorhergehenden berechnet. An diesen Algorithmus werden nun bestimmte Anforderungen gestellt, insbesondere soll er nicht bestimmte Werte "bevorzugen" und sich nicht zu schnell wiederholen. Diese Anforderungen werden in idealer Weise von der RND-Funktion erfüllt. Die RND-Funktion ist - wie Sie vielleicht wissen - eigentlich gar keine Zufallsfunktion, sondern führt folgende Rechnungen aus mit dem jeweils letzten Ergebnis, das zwischengespeichert wird:

- Multiplikation mit 11879546
- Addition von 0.000392767774
- Vertauschen der einzelnen Bytes der Zahl in Binärdarstellung

Da wir nun nur an Zahlen kleiner als 255 interessiert sind, und die Funktion durch Wahl ähnlicher Parameter sich bald wiederholt (immer noch erstaunlich selten, probieren Sie es einmal aus!), multiplizieren wir den Wert noch 10 und speichern ihn als solchen in einer Variablen, die wir das nächste Mal wieder als Parameter benutzen. Für die EOR-Funktion benutzen wir die von dem Wait-Befehl schon bekannte Formel:

```
a EOR b = (a OR b) and not (a AND b)
```

Jetzt haben wir alles zusammen, was wir brauchen, und gehen sofort an unser Demo-Programm heran:

```
10 INPUT "STARTWERT,TEXT",S,IN$:ME=S
20 C$="":FOR T=1 TO LEN(IN$)
30 S=RND(-S)*10:C=S*25.5+1:A=ASC(MID$(IN$,T,1))
40 C$=C$+CHR$((A OR C) AND NOT(A AND C))
50 NEXT T
60 PRINT "CODIERTER TEXT:",C$
70 IN$="":STOP
80 S=ME:IN$=C$:GOTO 20
```

Geben Sie nun einen Startwert und einen Text ein, und Sie werden nach kurzer Zeit völligen Unsinn auf dem Bildschirm angezeigt bekommen. Versuchen Sie einmal, aus dem Inhalt von C$ auf Ihren Text zurückzuschließen. Selbst bei genauer Kenntnis sowohl des Textes, des Startwertes als auch des Prinzips wird Ihnen dies nicht gelingen!

Wie Sie sehen, ist IN$ gelöscht, es scheint unmöglich, den ursprünglichen Text wiederzuerlangen. Einzig allein der für sich allein nicht sehr aussagekräftige Inhalt von C$ und der Startwert, der in ME zwischengespeichert ist, sind noch vorhanden. Durch "cont" erhalten Sie nun tatsächlich Ihren Text zurück.

Wenn Sie nun den Inhalt der Variablen C$ auf Diskette gespeichert hätten und dazu das Programm, nicht aber den Startwert, so könnte niemand außer Ihnen den eingegebenen Satz entschlüsseln, Sie müßten jedoch nur den Startwert eingeben. Dies war nun das Prinzip, realisieren müssen Sie es schon selber, hier nur noch ein paar Hinweise:

Eingangs sprach ich von "Paßwort". Tatsächlich können Sie statt des numerischen Startwertes einfach ein Paßwort abfragen. Dann addieren Sie die ASCII-Werte der einzelnen Buchstaben, und benutzen diese Summe als Startwert. Durch Abändern der Formel kann man den Algorithmus insbesondere für längere Texte noch sicherer machen: Rechnen Sie mit 16-Bit-Werten und fassen immer 2 Bytes zusammen, oder benutzen Sie gar 32- oder 64-Bit-Werte für 4 oder 8 Bytes. Auch kann ein zweiter Startwert (bzw. Paßwort) eingeführt werden, den Sie in die Formel einfließen lassen können (beispielsweise anstatt des *10 mit diesem 2. Wert multiplizieren).

8.11 Tastenwiederholfunktion

Wenn Sie beim C64 die Cursortasten drücken und gedrückt halten, so stellt sich nach kurzer Zeit ein Modus ein, wobei die Cursorbewegung wiederholt wird, solange die Taste gedrückt wird. Dies ist bei den "normalen" Tasten nicht der Fall. Von anderen Computern kennt man, daß alle Tasten eine Wiederholfunktion aufweisen.

Man mag dies als Vorteil oder Nachteil erachten, alle Argumente dafür oder dagegen weisen Sie als C64-Besitzer souverän von sich, denn bei Ihrem Computer können Sie selbst wählen, ob Sie die Tastenwiederholfunktion nur für die Cursortasten oder für alle Tasten aktiviert oder ganz ausgeschaltet haben wollen. Ferner können Sie einstellen, wie schnell der Wiederholmodus nach Beginn des Tastendrucks einsetzen soll und wie schnell nacheinander die einzelnen Tasten wiederholt werden sollen. Hier die entsprechenden Pokes:

POKE 650,n für	n=0	Wiederholfunktion nur für Cursor-Inst/Del- und Leertaste
	n=64	Wiederholfunktion ganz aus
	n=128	Wiederholfunktion für alle Tasten
POKE 651,n		Wiederholgeschwindigkeit
POKE 652,n:		Wiederholverzögerung, Zeit, die eine Taste gedrückt sein muß, bis Wiederholfunktion aktiviert wird

9. Spiele

Daß man mit dem C64 sehr gut spielen kann, werden die meisten von Ihnen schon gemerkt haben. Übrigens ist es keine Schande, mal ein Computergame zu spielen, was von manchen jedoch leider behauptet wird. Doch da hat man sich gerade das 'supercoolste' Spiel zugelegt, spielt eine Woche, doch plötzlich taucht im 13 Level so ein Ungeheuer auf, gegen das absolut kein Kraut (oder keine Waffe) gewachsen ist. Eine Möglichkeit besteht nun darin, das Spiel zur Seite zu legen und aufzugeben, oder - Sie haben ein Buch wie dieses.

Dieses Kapitel soll Ihnen einige Tips und Tricks zu verschiedenen Spielen zeigen und beim Überwinden von Problemen helfen. Am Schluß folgt dann noch eine POKE-Tabelle, wodurch Sie z.B. unsterblich werden (natürlich nur im Spiel) oder alle Feinde überwinden können. Also in diesem Sinne: Viel Spaß beim Spielen!

9.1 Spielhilfen

Dieser Abschnitt gibt Ihnen kurze Hilfen, wie Sie ein Spiel besser bewältigen können. Los geht's!

ACE OF ACES

Hat man einmal den ersten Platz erreicht, so ist es nicht mehr schwer, diesen zu halten. Geben Sie einfach als Namen DUSTY BUG ein, und Sie werden unsterblich.

ARKANOID

Wußten Sie schon, daß es in Arkanoid einen Trainer-Modus gibt? Stellen Sie am Anfang einmal folgende Parameter ein: 2 Players / 1 Joystick / 1 Device. Der erste Spieler hat nun ein ganz normales Spiel. Der zweite Spieler befindet sich im Trai-

ner-Modus, sobald er 20000 Punkte erreicht hat. Dann erhöht sich nämlich sein Lebenskonto bei jeder Kollision um ein Leben, bis man 87 Leben hat, die man auch nicht mehr verliert. Um Arkanoid durchzuspielen, muß man sich 2 Stunden Zeit nehmen. Viel Spaß!

BOUNTY BOB STRIKES BACK

Hier die Codes: ABC, IHB, LTO, MLB, DVJ, OAQ, PHH, XNR. Außerdem gibt es für dieses Spiel zwei verschiedene Arten, in den Cheat-Modus zu gelangen. Der erste besteht darin, 57502 als Code einzugeben und dann die F3-Taste und A gleichzeitig zu drücken. Die zweite besteht darin, als Option einfach F einzugeben.

BREAKTHRU

Hier gibt es einen Trick für den ersten Level. Drücken Sie direkt im ersten Level beim Losfahren die Space-Taste, und bewegen Sie den Steuerknüppel zweimal nach unten. Daraufhin befindet sich das Auto in der obersten Zeile, und Sie können den ersten Level ungeschoren durchfahren.

BRUCE LEE

Hier kann man ganz einfach 99 Leben bekommen: Gehen Sie zu den Tellern, bei denen man normalerweise 1 Leben bekommt. Berühren Sie diese, und Sie haben 99 Leben zur Verfügung. Dieser Trick funktioniert jedoch nur im 2-Player-Modus.

BUBBLE BOBBLE

Wenn Sie alle drei Leben verloren haben, so ist das Spiel noch längst nicht zu Ende. Drücken Sie einfach solange den Joystickknopf, bis Sie wieder drei neue Leben erhalten.

CAVERN OF RICHES

Hier ein paar Zauberwörter für dieses Adventure: AWAY, XYZZY, EGYPT und letztendlich noch SESAME.

DEFLEKTOR

Bei Deflektor gibt es einen recht einfachen Trick. Drücken Sie einfach die < + >- oder < - >-Tasten, um in die verschiedenen Level zu kommen.

EXOLON

Hier gelangen Sie in den Cheat-Modus, indem Sie im Menü den Menüpunkt 'Tasten umdefinieren' anwählen. Geben Sie nun das Wort ZORBA ein, und Sie haben unendlich viele Leben.

GREAT GIANNA SISTERS

Bei diesem süßen Hüpfspiel kommt man einen Level weiter, indem man den Namen des Programmiers eingibt und die Tasten gedrückt hält

GREEN BERET

Zwei Tricks: Am Ende des ersten Levels sollten Sie sich auf den Boden legen und immer den Feuerknopf drücken. Dadurch werden alle Gegner besiegt. Erledigen Sie immer erst die Leute, die von hinten kommen, da man deren Schüssen schwerer ausweichen kann.

KARATE KID II

Durch das Drücken der Taste P gelangt man einen Level weiter.

LAP OF THE GODS

Hier gelangt man in den Cheat-Modus, indem man im High-Score einfach CHEAT eingibt.

LEGENT OF SINDBAD

Auch hier gibt es einige Codes: COSMO, STROM, TWIST.

LEGIONS OF DEATH

Kaufen Sie Schiffe während der Schlachten, denn dann wird Ihnen kein Geld dafür abgeknöpft!

NEBULUS

Auch hier gibt es einen Trainer-Modus. Aktivieren Sie den Pause-Modus. Drücken Sie nun die Tasten <Hochpfeil>, <Linkspfeil> und den Buchstaben J gleichzeitig. Sie haben nun unendlich viel Zeit. Außerdem kann man noch durch Drücken der Zahlen 1-8 die verschiedenen Level anwählen.

NEMESIS

Hier gibt es vier Cheat-Levels. Den ersten erreichen Sie, indem Sie im zweiten Level die Steinschlaufe, die sich am Ende des Levels befindet, aufschießen und sie dann betreten. Der zweite Cheat-Level ist im dritten Level zu erreichen. Dort gibt es zwei Figurenpaare, die Rücken an Rücken stehen. Zerstören Sie beim zweiten Paar die erste Figur, und Sie gelangen wieder in einen Cheat-Modus. Der dritte ist dadurch zu erreichen, daß Sie im siebten Level kurz vor dem Ende in die Öffnung des Ovals fliegen. Den vierten Cheat-Modus erreichen Sie durch Drücken der Taste <SHIFT/LOCK>. Viel Spaß bei der zusätzlichen Punktejagd! Das ist jedoch nicht das einzige, was ich zu Nemesis anzubieten habe. Es gibt in diesem Spiel nämlich auch einen Trainer-Modus. Drücken Sie einfach während des Fluges des Tasten JKL gleichzeitig.

OBLIVION

Hier gelangt man in den nächsten Level durch Drücken der Tasten 1, 2, Z, X, C und V.

ONE MAN AND HIS DROID

Hier die Codes für dieses Spiel

NONE, BUBBLE, COMMODORE, FINDERS, GENETIC, ZAPPED, TIMEWARP, ECTOPLASM, GORGEOUS, SEASIDE, GIZMO, KING KONG, HOLOGRAM, CURRYRICE, COFFEE, CASSETTE, TELESCOPE, COMPUTER, EDACRAEDA, ALICE.

OUT RUN

Auch hier gibt es einen Cheat-Modus. Wenn Sie einen Unfall gebaut haben (natürlich nur auf dem Bildschirm), so sollten Sie mit dem Joystick rütteln. Daraufhin fetzt der Wagen wieder los. Haben Sie die Höchstgeschwindigkeit erreicht, so wird die Sprite-Kollision außer Kraft gesetzt.

PARALLAX

Hier die Codes: STACK, JEVEL, PARCH, SALON, GLOBE.

RAID OVER MOSCOW

Bei RAID OVER MOSCOW kann man die verschiedenen Sequenzen einzeln zum Üben anwählen. Drücken Sie dazu die Taste <RUN/STOP-RESTORE> mit Q (Raketensilos), <Pfeil Links> (Kreml), 1 (Reaktorraum), 2 (Schlußszene). Dieser Trick funktioniert aber nur, wenn die Weltkarte gerade angezeigt wird.

REVENGE OF MUTANT KAMELS

Bei diesem Spiel müssen Sie nach dem Programmstart einfach das Wort GOATS eingeben, und Sie befinden sich im Cheat-

Modus. Es ist sogar möglich, durch Drücken einer bestimmten Taste zwischen den Bildern hin- und herzuspringen. Diese Taste müssen Sie jedoch selbst rausfinden (Tip: Sie ist nicht zu übersehen)!

ROAD RUNNER

Sie können einen Level überspringen, indem Sie einfach vor die Kiste am oberen Bildrand fahren.

SPINDIZZY

Auch in diesem Spiel gibt es einen Cheat-Modus! Geben Sie einfach nach dem Laden die drei Buchstaben PAT ein, und Sie haben unbegrenzte Zeit.

SUPER ALIENS

Da der Anflug recht schwer ist, haben die Programmierer einen Schummelmodus eingebaut. Geben Sie einfach als Missions-Code die Zeichenfolge 2727H ein, und der Landeanflug wird Ihnen erspart.

SUPERSTART ICE-HOCKEY

Wußten Sie, daß es einen Trick zum Tore-Schießen gibt? Schießen Sie einfach von weiterer Entfernung aufs Tor. Sobald der Puck in der Luft ist, drücken Sie die Restore-Taste. Danach sollten Sie die Space-Taste drücken - und siehe da, der Ball ist (meistens wenigstens) im Tor!

THE GREAT GURIANOS

Hier gelangt man einen Level weiter, indem man folgende Tasten gleichteitig drückt: M, <Komma>, ., <Cursor Rechts>, <Cursor Links>. Danach sollten Sie noch die Return-Taste betätigen (sofern Sie noch einen Finger frei haben).

TIGER MISSION

Hier gelangt man durch Drücken der Tasten 2, Q, R, L, I, K, <CRTL>, <COMMODRE> in den Cheat-Modus (für gebrochene Finger bin ich nicht verantwortlich!).

TRANTOR

Hier nur ein kurzer Hinweis: Die Code-Wörter haben etwas mit dem Bereich Computer zu tun!

TWINLY

Drücken Sie im Titelbild folgende Tasten gleichzeitig: 1, 2, 3, 4, D, <CRTL> ,<RUN/STOP>, <Pfeil Links>. Sie können nun mittels des Joysticks in Port 2 die verschiedenen Level auswählen.

ZAXXON

Auch hier gibt es einen Trainer-Modus. Geben Sie im Titelbild einfach das Wort RED ein, und Sie haben nun unendlich viele Raumschiffe zur Verfügung.

9.2 Spiele-Pokes

Wie die Überschrift dieses Kapitels schon sagt, werde ich Ihnen hier Pokes verraten, die viel bewirken können. Laden Sie - falls nicht anders gesagt - das Spiel, starten Sie es und führen Sie dann einen Reset aus. Geben Sie danach die Pokes immer in einer Zeile ein, und starten Sie das Spiel wieder durch den angegebenen SYS-Befehl. Die Pokes sind natürlich ohne Gewähr. Hier nun die versprochenen Pokes:

ANTIRIAD

```
Poke 35486,165 : Poke 35496,16 : Sys 2080
```

APOLLO 18

Poke 2356,X : Sys 2335

Für X kann man Werte zwischen 1 und 11 eingeben, wodurch sich dann die verschiedenen Spielsequenzen ergeben.

ARCANA

Poke 12933,0 : Poke 12934,2 : Sys 4096

BACK TO REALITY

Poke 20109,173 : Poke 27337,96 : Sys 16384

BOMB JACK II

Poke 10715,234 : Poke 10716,234 : Poke 10717,234 : Sys 15146

BREAKTHRU

Poke 6604,173 : Sys 2560

BUGGYBOY

Poke 4768,133 : Poke 47769,20 : Poke 39927,96 : Poke 2048,32 : Poke 2049,104 : Poke 2050,13 : Sys 2560

Durch diese Pokes können Sie nun den Kurs auswählen

CRAZY COMETS

Poke 36888,120 : Sys 26386

DRUID

Poke 39271,255 : Sys 5120

ELEVATOR ACTION

For a=50911 to 50915: Poke a,234 : Next : Sys 53200

EXOLON

Poke 7427,205 : Sys 2061

FIRETRACK

Poke 12285,234 : 12286,234 : Poke 12287,234 : Sys 9216

FIRETRAP

Poke 7500,234 : Poke 7501,234 : Sys 4096

GALVAN

Poke 30602,234 : Poke 30603,234 : Poke 30603,234 : Sys 12288

GAME OVER

Poke 15244,234 : Poke 15245,234 : Sys 2304

GREAT GIANNA SISTERS

Poke 2446,255 : Poke 6697,255 : Sys 2098

GYROSCOPE

Poke 46687,76 : Poke 46688,105 : Poke 46689,182 : Sys 2067

HADES NEBULA

Poke 2279,X (Leben) : Sys 18550

HE-MAN

Poke 12651,234 : Poke 12652,234 : Poke 12653,234Poke 17610

INTO THE EAGLES NEST

Poke 24651,234 : Poke 26542,234 : Poke 26543,234 : Sys 32784

JAIL BREAK

For a=52050 to 52052:Poke a,234:Poke a+47,234:Next : Sys 1200

KRACKOUT

Poke 32934,0-100 (Levelangabe) : Sys 32837

LIGHT FORCE

Poke 11547,5 : Sys 6713

LIVING STONE

Poke 29744,173 : Poke 31026,173 : Sys 16384

MERMAID MADNESS

Poke 17274,169 : Poke 17275,0 : Poke 17276,236 : Sys 16384

MUTANTS

Poke 9273,230 : Sys 4096

NEMESIS

Poke 5975,234 : Poke 5976,234 : Poke 5977,234 : Sys 5779

PANTHER

Poke 14127,169 : Sys 4096

PARALLAX

Poke 5796,96 : Sys 319

P.O.D

Poke 9467,173 : Sys 7936

RANARAMA

Poke 37104,96 : Poke 33969,234 : Poke 33970,234 : Sys 32768

SORCERY

Poke 56325,255 : Sys 31744

SPACE HARRIER

Poke 5834,234 : Poke 5834,234 : Poke 5836,234 : Sys 2128

TERRA COGNITA

Poke 26703,255 : Sys 24576

THE SENTINEL

Poke 6679,173 : Poke 8512,10 : Sys 16128

THRUST

Poke 6139,234 : Poke 6140,234 : 6141,234 : Sys 2304

TRAPDOOR

Poke 14914,96 : Sys 14336

WARHAWK

Poke 27090,234 : Poke 27091,234 : Poke 27092,234 : Sys 24604

WIZBALL

Poke 37052,0 : Poke 65303,0 : Poke 65356,0 : Poke 65395,0 : RUN

XEVIOUS

Poke 5605,76 : Poke 5606,31 : Poke 5607,X (Leben) : Sys 5000

10. GEOS-Anwendungen

In diesem Kapitel möchte ich Ihnen ein paar sinnvolle Anwendungsmöglichkeiten mit GEOS aufzeigen. Sollten Sie in GEOS schon fit sein, sehen Sie mir bitte nach, daß die ersten beiden Beispiele "Visitenkarten" und eine "Speisekarte" sehr ausführlich beschrieben sind. Aber ich habe hierbei an die vielen Anwender gedacht, die mit GEOS noch nicht so intensiv gearbeitet haben. Sollten Sie zu den GEOS Neulingen gehören, arbeiten Sie die ersten beiden Anwendungen intensiv durch. Ich führe Sie hier Schritt für Schritt vom leeren Zeichenblatt zur fertigen Anwendung. Allen GEOS-Neulingen sei gesagt, daß nach der Eingabe von

```
LOAD"GEOS",8,1 <RETURN>
```

das Eingeben von Befehlen über die Tastatur der Vergangenheit angehört, da Sie mit GEOS eine grafische Benutzeroberfläche zur Verfügung haben, die die Möglichkeiten des Rechners voll ausnutzt. Sie werden sehen, daß Sie innerhalb kurzer Zeit recht gut mit den Besonderheiten von GEOS zurechtkommen und schnell wissen, wie man zu seinem gewünschten Ziel kommt.

Ich werde bei der Beschreibung der einzelnen Beispiele die gravierenden Punkte der jeweiligen Anwendung hervorheben. Ich arbeite übrigens mit der GEOS-Version 1.3 deutsch. Daher werde ich natürlich auch die hier benutzten Befehlsworte benutzen. Sollten Sie noch eine amerikanische Version benutzen, fällt es Ihnen sicher nicht schwer, die entsprechenden Befehle anzuwählen.

So, genug der Vorrede. Sie haben Ihre GEOS-Bootdiskette schon im Laufwerk und Ihre Arbeitsdiskette vorbereitet? In der Zeit, in der GEOS geladen wird, sollten Sie sich den benötigten Inhalt der Arbeitsdiskette für die folgenden Anwendungsbeispiele ansehen, um eventuell das eine oder andere zu löschen bzw. fehlende Programmteile dazuzukopieren.

10.1 Inhalt der Arbeitsdisketten

Ihre Arbeitsdiskette sollte folgende Programme enthalten. So haben Sie immer alles, was sie brauchen, auf einer Diskette.

- Desktop
- Geowrite
- Geopaint
- Voreinstellung
- Druckertreiber
- Schriftfonts
  - cory ge
  - Dwinelle ge
  - California ge
  - Roma ge

10.2 Voreinstellung

Bei der Arbeit mit GEOPAINT ist es besonders beim punktgenauen Zeichnen ratsam, die Anfangsgeschwindkeit des Mauspfeils zu reduzieren, da man sonst ständig am gewünschten Punkt vorbeifährt und sich somit unnötige Arbeit macht. Aber man kann das Programm VOREINSTELLUNG ja auch aus GEOPAINT und GEOWRITE laden, um die Einstellungen zu korrigieren. Nehmen Sie sich ruhig Zeit, die Uhr und das Datum richtig zu stellen. Denken Sie daran, das jede Grafik und jeder Text mit den eingestellten Daten gespeichert wird. Sie haben hiermit jederzeit einen Überblick, was Sie wann gemacht haben.

10.3 Speisekarte

In diesem Kapitel möchte ich Ihnen zeigen, wie Sie mit GEOPAINT eine Speisekarte gestalten können. Ich habe Ihnen hier die einzelnen Schritte aufgezählt, so daß sie Schritt für Schritt nachvollzogen werden können.

- **Geopaint-Icon doppelklicken**
  Geopaint wird geladen.
  Das Auswahlfenster erscheint.

- **Option NEUES DOKUMENT ERSTELLEN anwählen**
  Frage nach Dokumentname

- **Name "SPEISEPLAN" eingeben und mit Return bestätigen.**
  Geopaint-Zeichenblatt erscheint auf dem Bildschirm.

- **Werkzeugleiste / Linienfunktion anwählen**
  Liniensymbol wird invers dargestellt.
  Untermenü Maßeinheiten erscheint unten links.

- **Cursor im Arbeitsblatt nach oben links bewegen. Anfangspunkt in Höhe des Symbols "Linien" am linken Blattrand einmal klicken.**
  Linie nach rechts oben (x13,y13).
  Bewegungen in x- und y-Richtung werden im Hilfsmenü angezeigt.

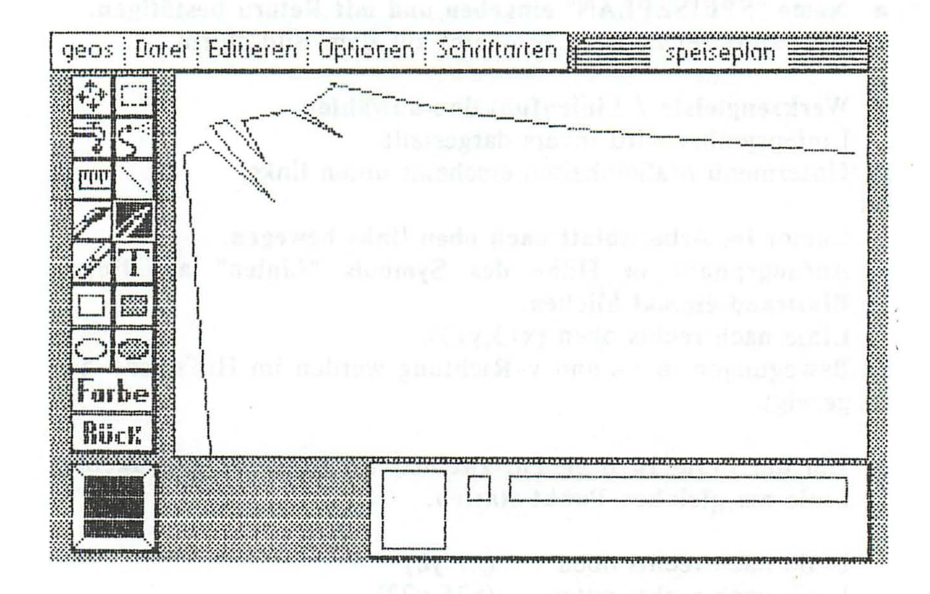
- **Um die Linie zu beenden, zweimal klicken, und die nächste Linie am gleichen Punkt starten.**

Linie nach rechts oben	(x7 y8)
Linie nach rechts unten	(x25 y22)
Linie nach links oben	(x24 y29)
Linie nach rechts oben	(x5 y2)
Linie nach rechts unten	(x14 y17)
Linie nach links oben	(x8 y17)
Linie nach rechts oben	(x18 y3)
Linie nach rechts unten	(x21 y11)
Linie nach links oben	(x15 y11)
Linie nach rechts oben	(x11 y11)
Linie nach rechts unten	(x35 y10)
Linie nach rechts unten	(x55 y7)
Linie nach rechts unten	(x77 y7)
Linie nach rechts unten	(x36 y2)

Rechter Rand erreicht.

- **Jetzt vom Anfangspunkt der Linien rechts neben dem Liniensymbol der Werkzeugleiste "Linie" nach rechts unten (x10 y112).**

Hier nun eine Abbildung, die zeigt, wie es ungefähr aussehen sollte:



- **Werkzeugleiste / Rahmen anwählen**
  doppelklicken
  Ganzer Bildausschnitt wird gerahmt

- **Menüleiste / Editieren / Kopieren anwählen**
  Gerahmter Bildausschnitt wird zwischengespeichert.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt auf rechte obere Bildecke bewegen.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**
  Werkzeugleiste / Rahmen doppelklicken
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben**
  Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln x**
  Bild wird um die x-Achse gedreht.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt auf rechte untere Bildecke bewegen.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**
  Werkzeugleiste / Rahmen doppelklicken
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben**
  Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln x /Spiegeln y**
  Bild wird erst um die x-Achse und dann um die y-Achse gedreht.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt auf rechte untere Bildecke bewegen.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben**
  Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln y**
  Bild wird um die y-Achse gedreht.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt unten in die Mitte des Bildes bewegen.
  klicken
  Linien ragen von beiden Seiten in den Bildausschnitt.
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

- **Werkzeugleiste / Linien anwählen**
  Cursor links an der Linie ansetzen
  klicken
  Linienfunktion ist aktiviert.
  Linie nach rechts oben (x34 y 2)
  Linie nach rechts (x51 y 0)
  Linie nach rechts unten (x34 y 2)

- **Werkzeugleiste / Rahmen anwählen**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Kopieren anwählen**
  Gerahmter Bildausschnitt wird zwischengespeichert.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt ganz nach oben bewegen.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben**
  Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln y**
  Bild wird um die y-Achse gedreht.

- **Menüleiste / Datei / aktualisieren**
  Das derzeitige Bild wird gespeichert.

- **Menüleiste / Datei / Übersicht**
  Es wird jetzt das gesamte Bild auf dem Bildschirm gezeigt.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt von ganz links oben runterbewegen, bis nur ein kleiner Rest der Linie zu sehen ist.

- **Werkzeugleiste / Linien anwählen**
  Cursor an der Linie ansetzen.
  Linie nach rechts unten (x10 y112)

- **Werkzeugleiste / Rahmen anwählen**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Kopieren anwählen**
  Gerahmter Bildausschnitt wird zwischengespeichert.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt ganz nach rechts bewegen.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben anwählen**
Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln x**
Bild wird um die x-Achse gedreht.

- **Werkzeugleiste / Bewegungspfeile anwählen**
Bildausschnitt nach unten bewegen, bis der Anfang der Linie ins Bild kommt.
klicken
Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

- **Werkzeugleiste / Rahmen doppelklicken**
Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben anwählen**
Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln x / Spiegeln y**
Bild wird erst um die x-Achse und dann um die y-Achse gedreht.

- **Werkzeugleiste / Bewegungspfeile anwählen**
Bildausschnitt ganz nach links bewegen.
klicken
Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

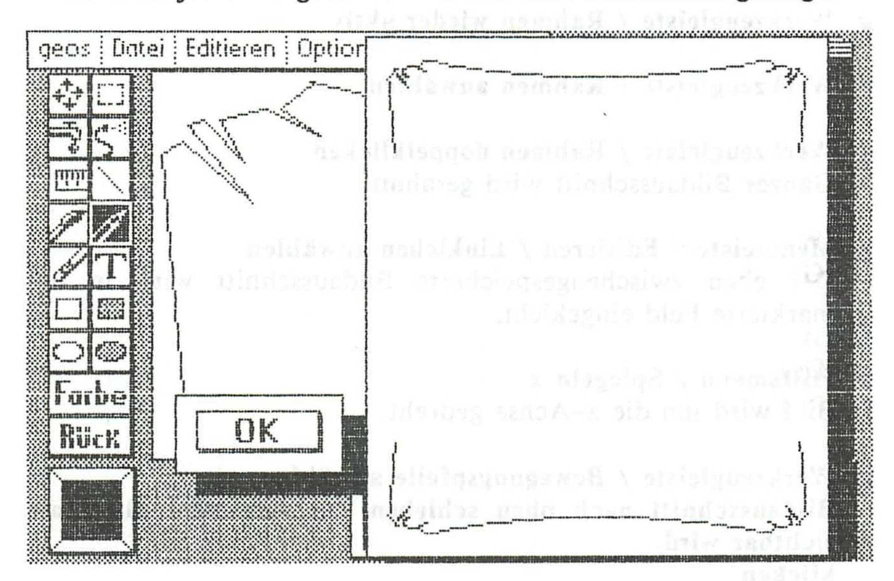
- **Werkzeugleiste / Rahmen doppelklicken**
Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben anwählen**
Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln y**
  Bild wird um die y-Achse gedreht.

- **Menüleiste / Datei / aktualisieren**
  Das derzeitige Bild wird gespeichert.

- **Menüleiste / Datei / Übersicht**
  Es wird jetzt das gesamte Bild auf dem Bildschirm gezeigt.



- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt nach oben bewegen, bis nur der Rest der Linie zu sehen ist.

- **Werkzeugleiste / Linien anwählen**
  Cursor an der Linie ansetzen.
  Linie nach rechts oben (x7 y90)
  Linie nach rechts oben (x0 y20)

- **Werkzeugleiste / Ramen anwählen**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Kopieren anwählen**
  Gerahmter Bildausschnitt wird zwischengespeichert.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschitt ganz nach rechts bewegen.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

- **Werkzeugleiste / Rahmen anwählen**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben anwählen**
  Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln x**
  Bild wird um die x-Achse gedreht.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt nach oben schieben, bis von oben die Linie sichtbar wird.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben anwählen**
  Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln x / Spiegeln y**
  Bild wird erst um die x-Achse und dann um die y-Achse gedreht.

- **Werkzeugleiste / Linienfunktion anwählen**
  Cursor unten an der Linie ansetzen.
  Linie nach unten rechts (x1 y16)

- **Werkzeugleiste / Rahmen anwählen**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Kopieren anwählen**
  Gerahmter Bildausschnitt wird zwischengespeichert.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt ganz nach links bewegen.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

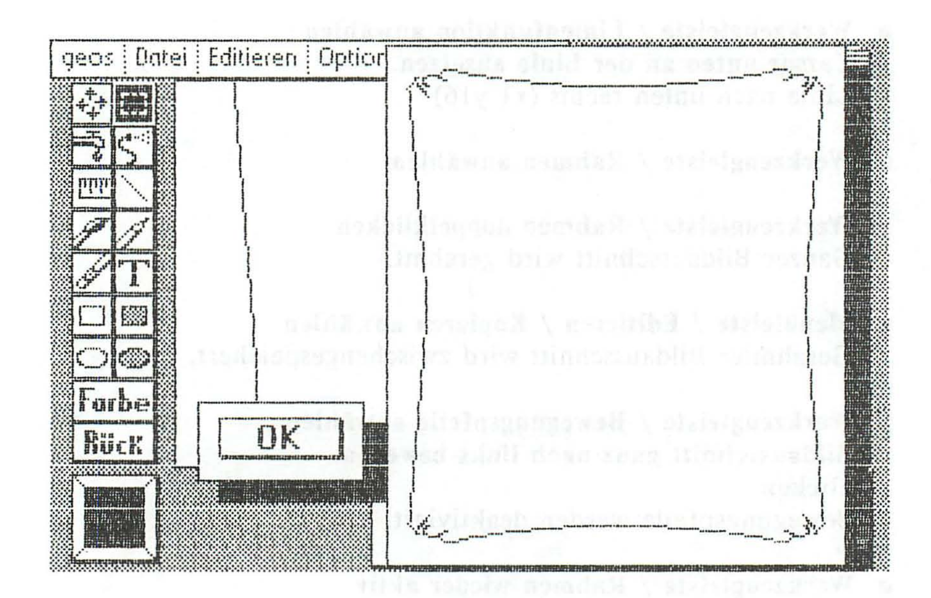
- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben anwählen**
  Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Hilfsmenü / Spiegeln x**
  Bild wird um die x-Achse gedreht.

- **Menüleiste / Datei / Aktualisieren**
  Das derzeitige Bild wird gespeichert.

- **Menüleiste / Datei / Übersicht**
  Es wird jetzt das gesamte Bild auf dem Bildschirm gezeigt.



- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt erst in die obere linke Ecke bewegen und anschließend soweit nach rechts und nach unten, daß man den Rahmen nicht mehr sieht.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen wieder aktiv**

- **Werkzeugleiste / Text anwählen**
  Den Cursor in waagerechter Verlängerung unter das Rahmenfeld und in senkrechter Verlängerung zwischen "Editieren" und "Optionen" der Menüleiste bewegen.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster mit dem Joystick nach rechts unten aufziehen.**
  klicken
  TextCursor erscheint im Textfenster.

- **Menüleiste / Schriftarten / Roma 24 Point anwählen**

- **Hilfsmenü /Fett / Kursiv / Kontur anwählen**
  Text eingeben "Zum wilden"

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Nach rechts schieben, bis die Schrift auf der linken Seite fast verschwunden ist.
  Rechts neben der Schrift ein neues Textfenster öffnen.
  Text "Bären"

- **Menüleiste / Datei / aktualisieren**
  Das derzeitige Bild wird gespeichert.

- **Menüleiste / Datei / Übersicht**
  Es wird jetzt das gesamte Bild auf dem Bildschirm gezeigt.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt ganz links unter die Schrift bewegen.

- **Werkzeugleiste / Linie anwählen**
  Cursor in die Bildmitte.
  Linie nach rechts bis zum Bildschirmrand ziehen.

- **Werkzeugleiste / Lineal anwählen**
  60 Punkte von der Linie nach unten messen.

- **Werkzeugleiste / Linie anwählen**
  60 Punkte unter der Linie eine neue Linie parallel zur oberen Linie ziehen.

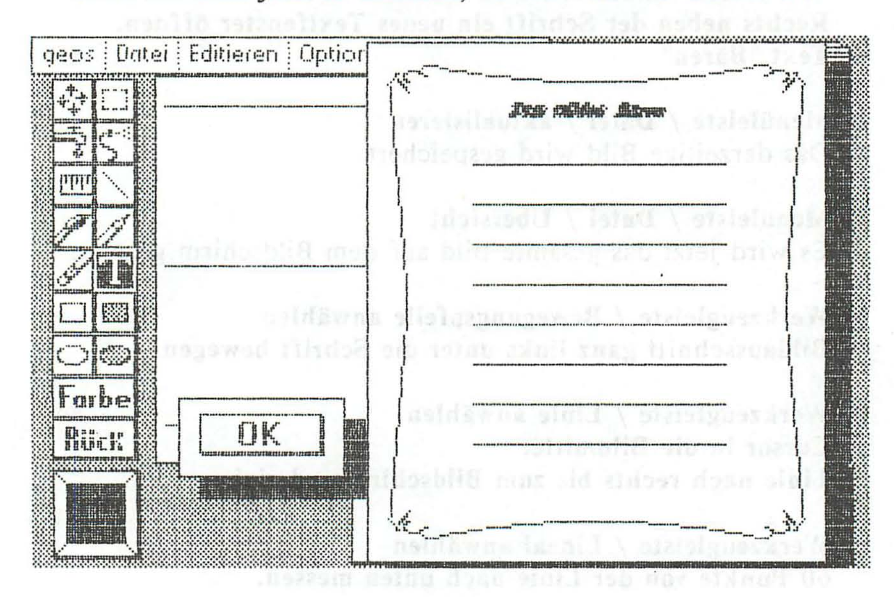
- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildfenster nach rechts schieben, bis von den zwei Linien auf der linken Seite nur noch die Enden zu sehen sind.

- **Werkzeugleiste / Linie anwählen**
  Die Linien über den ganzen Bildausschnitt verlängern.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Im Abstand von jeweils 60 Punkten noch 6 Linien untereinander ziehen.

- **Menüleiste / Datei / aktualisieren**
  Das derzeitige Bild wird gespeichert.

- **Menüleiste / Datei / Übersicht**
  Es wird jetzt das gesamte Bild auf dem Bildschirm gezeigt.
  Wenn Ihr Bild jetzt so aussieht, können wir weitermachen:



- **Menüleiste / Bewegungspfeile anwählen**
  Bildausschnitt nach unten links schieben, bis oben die letzte Linie und unten die Endbegrenzung sichtbar ist.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Text anwählen**
  Den Cursor in waagerechter Verlängerung des Bleistiftfeldes und in senkrechter Verlängerung unter das Feld Editieren bewegen.
  klicken
  Das Textfenster wird aktiviert.

  Textfenster mit dem Joystick nach rechts unten aufziehen, Ca. 3 cm (die untere Linie darf nicht in das Textfenster einbezogen werden).

klicken
Textcursor erscheint im Textfenster.

- **Menüleiste / Schriftarten / Roma / 18 Punkte anwählen**

- **Hilfsmenü / normal anwählen**
  Text eingeben "Bisonallee 111, 8000 München".

- **Menüleiste / Bewegungspfeile anwählen**
  Bildschirmausschnitt nach rechts schieben, bis das Wort München fast verschwunden ist.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Text anwählen**
  klicken
  Das Textfenster wird aktiviert.

- **Vom letzten Buchstaben von München nach rechts bis zum Rand und ca 3 cm nach unten öffnen.**
  klicken
  Textcursor erscheint im Textfenster.

  Text eingeben "22, Tel. 7654321".

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt verschieben, daß der Anfang der obersten Linie links unten im Ausschnitt ist.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Text anwählen**
  Cursor zwischen die Felder der Werkzeugleiste "Farbe" und "Kreis" bewegen.
  klicken
  Das Textfenster wird aktiviert.

- **Das Textfeld bis zum rechten Rand öffnen.**
  klicken
  Textcursor erscheint im Textfenster.

- **Menüleiste / Schriftarten / Roma / 18 Punkte anwählen**

- **Hilfsmenü / Fett + Kursiv anwählen**
  Texteingabe "Rehrücken"

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt verschieben, bis die Schrift links fast verschwunden ist.

- **Werkzeugleiste / Text anwählen**
  Cursor rechts neben das Wort Rehrücken bewegen, in Höhe oberer Rand der Kleinbuchstaben.
  klicken
  Das Textfenster wird aktiviert.

- **Das Textfeld bis zum rechten Rand öffnen (aufpassen, daß keine Linien überdeckt werden, da diese sonst anschließend verschwunden sind).**
  klicken
  Textcursor erscheint im Textfenster.

- **Menüleiste / Schriftarten / Roma / 12 Punkte anwählen**

- **Hilfsmenü / normal anwählen**
  Text "mit Kroketten, Rotkraut und Salat"

- **Werkzeugleiste / Text anwählen**
  Cursor rechts neben den Text am Ende der Linie bewegen.
  klicken
  Das Textfenster wird aktiviert.

- **Das Textfeld bis zum rechten Rand öffnen (Aufpassen: Keine Linien überdecken, da diese sonst verschwinden.**
  klicken
  Textcursor erscheint im Textfenster.

- **Menüleiste / Schriftarten / Roma / 24 Punkte anwählen**

- **Hilfsmenü / Fett + Kursiv anwählen**
  Text eingeben "49.- DM"

Verfahren Sie bei den nächsten entsprechend.
Menü und Preis frei wählbar
Fertig könnte es so aussehen:

Zum wilden Bären

Rehrücken mit Kroketten, Rotkraut und Salat *49,- DM*

Hasenbraten mit Kroketten, Rotkraut und Salat *49,- DM*

Wildschwein mit Kroketten, Rotkraut und Salat *49,- DM*

Bärenkeule mit Kroketten, Rotkraut und Salat *59,- DM*

Bisonsteak mit Kroketten, Rotkraut und Salat *59,- DM*

Büffellende mit Kroketten, Rotkraut und Salat *69,- DM*

Bisonallee 111, 8000 München 22, Tel. 7654321

10.4 Visitenkarten

Auch hier habe ich Ihnen noch einmal die einzelnen Arbeitsschritte zur Herstellung einer Visitenkarte aufgeführt.

- **Geopaint-Icon doppelklicken**
  Geopaint wird geladen
  Auswahlfenster erscheint

- **Option NEUES DOKUMENT ERSTELLEN anwählen.**
  Frage nach Dokumentname
  Name "VISITENKARTE" eingeben und mit Return bestätigen.
  Geopaint Zeichenblatt erscheint auf dem Bildschirm.

- **Werkzeugleiste / Text anwählen**
  Hilfsmenü Style erscheint unten links.

- **Menüleiste / Schriftarten / Roma / 24 Punkte anwählen**

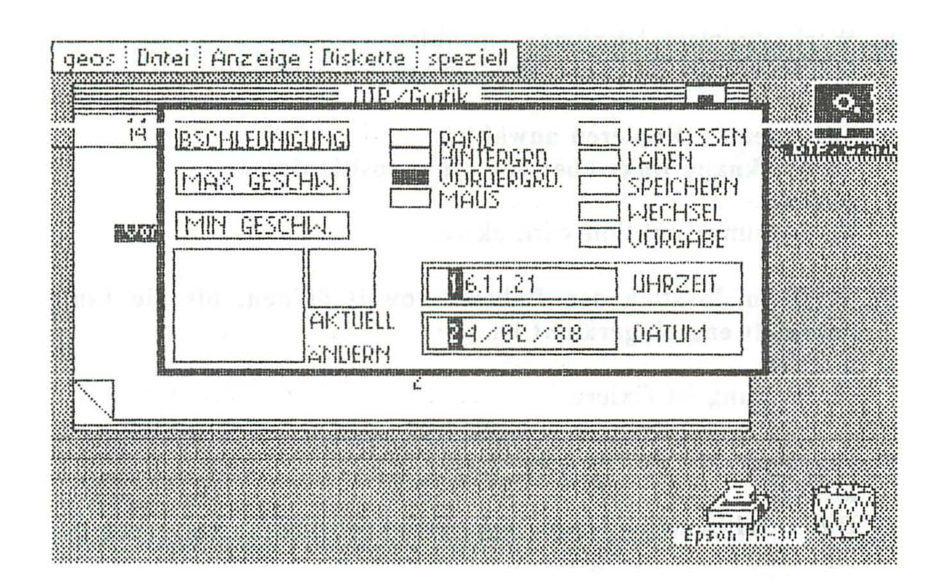
- **Hilfsmenü / Fett anwählen**
  Den Cursor bis fast in die obere linke Ecke bewegen.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster mit dem Joystick nach rechts unten öffnen.**
  klicken
  Textcursor erscheint im Textfenster.
  Text eingeben "Manfred Müller".

- **Werkzeugleiste / Linien anwählen**
  Hilfsmenü ändert sich von Style auf Maßeinheiten.

- **Cursor ca. 1 cm unter das M von Manfred bewegen.**
  klicken
  Linienfunktion wird aktiv.

- **Waagerechte Linie mit dem Joystick bis an den rechten Rand ziehen.**



- **Werkzeugleiste / Text anwählen**

- **Menüleiste / Schriftarten / Roma / 12 Punkte anwählen**

- **Hilfsmenü / Fett anwählen**
  Cursor in die Bildausschnittmitte bewegen.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster mit dem Joystick nach rechts unten öffnen.**
  klicken
  Textcursor erscheint im Textfenster.

- **Text eingeben "Mühlenstraße 13".**
  RETURN drücken, um einen Zeilenvorschub durchzuführen und in die nächste Zeile zu gelangen.
  Text eingeben "4000 Düsseldorf".
  2mal Return (2mal Zeilenvorschub, damit man eine Leerzeile erhält.

- **Text eingeben "Telefon 0211/654321"**

- **Werkzeugleiste / Rahmen anwählen**
  Hilfsmenü ändert sich von Style auf Editiermodus.

- **Hilfsmenü / Kopieren anwählen**
  Cursor knapp links über der Linie positionieren.
  klicken
  Markierungsfunktion wird aktiv.

- **Mit dem Joystick den Rahmen soweit öffnen, bis die Linie komplett eng eingerahmt ist.**
  klicken
  Markierung ist fixiert.

- **Cursor in den Rahmen bewegen.**
  klicken
  Die im Hilfsmenü gewählte Funktion ist aktiv.
  Der Cursor ist hell gefärbt.

- **Mit dem Joystick den Cursor zwischen Ort und Telefon bewegen.**
  Der markierte Rahmen folgt dem Cursor.
  klicken
  Die kopierte Linie ist fixiert.

- **Werkzeugleiste / Bleistift anwählen**
  Den Bleistift mit dem Joystick ganz an den linken Rand setzen. Es muß genau sein, da aus dem Bleistift sonst wieder der Mauspfeil wird.
  klicken
  Der Bleistift wird aktiv.

- **Bewegen Sie den Stift mit dem Joystick ganz nach oben. Wenn der Stift jetzt zum Mauspfeil wird, ändert es nichts an der Funktion. Bewegen Sie den inzwischen entstandenen Mauspfeil einmal um das ganze Arbeitsblatt herum. Der Bildausschnitt wird damit komplett eingerahmt.**

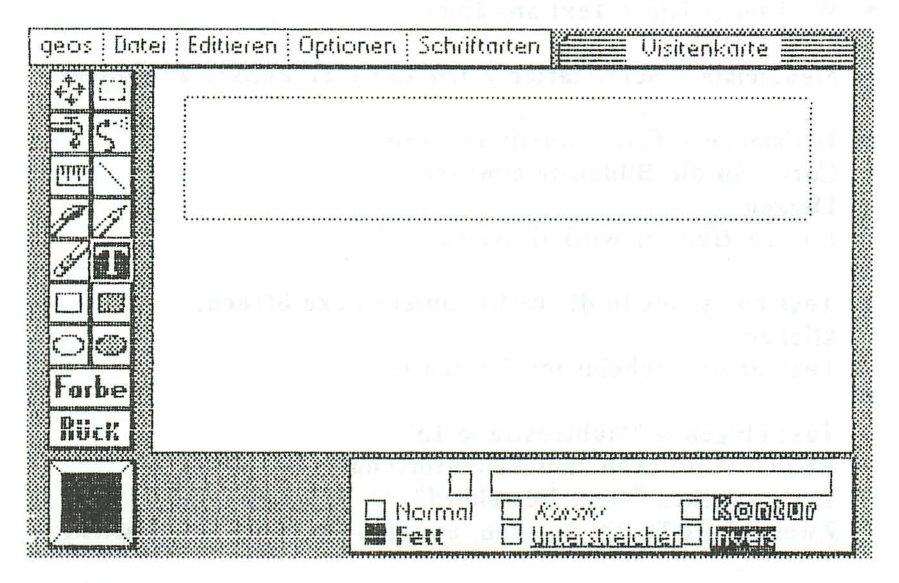
- **Werkzeugleiste / Bewegungspfeile anwählen**
  Mit dem Joystick den Bildausschnitt soweit nach rechts verschieben, bis der Bildschirm wieder leer ist.

- **Werkzeugleiste / Text anwählen**

- **Menüleiste / Schriftarten / LW Cal / 18 Punkte anwählen**

- **Hilfsmenü / Fett / Kursiv / Kontur anwählen**
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster von oben links anfangend über die ganze Breite öffnen.**



klicken
Textcursor erscheint im Textfenster.
Text eingeben "Manfred Müller"
Zwischen Vor- und Nachname 3 Leerzeichen

- **Werkzeugleiste / Text anwählen**

- **Menüleiste / Schriftarten / LW Cal / 12 Punkte anwählen**

- **Hilfsmenü / normal / Fett anwählen**
  Nach dem Anwählen des Punktes Normal werden alle vorher getroffenen Einstellungen gelöscht.

- **Cursor unter das M von Manfred positionieren.**
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster über die ganze Breite öffnen.**
  klicken
  Textcursor erscheint im Textfenster.

- **Text eingeben "Versicherungen aller Art".**

- **Werkzeugleiste / Text anwählen**

- **Menüleiste / Schriftarten / LW Cal / 12 Punkte anwählen**

- **Hilfsmenü / Fett / Kursiv anwählen**
  Cursor in die Bildmitte bewegen.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster bis in die rechte untere Ecke öffnen.**
  klicken
  Textcursor erscheint im Textfenster.

- **Text eingeben "Mühlenstraße 13"**
  RETURN drücken zum Zeilenvorschub.
  Text eingeben "4000 Düsseldorf".
  Zweimal RETURN drücken, um eine Leerzeile zu erhalten.

- **Text eingeben "0211/654321"**

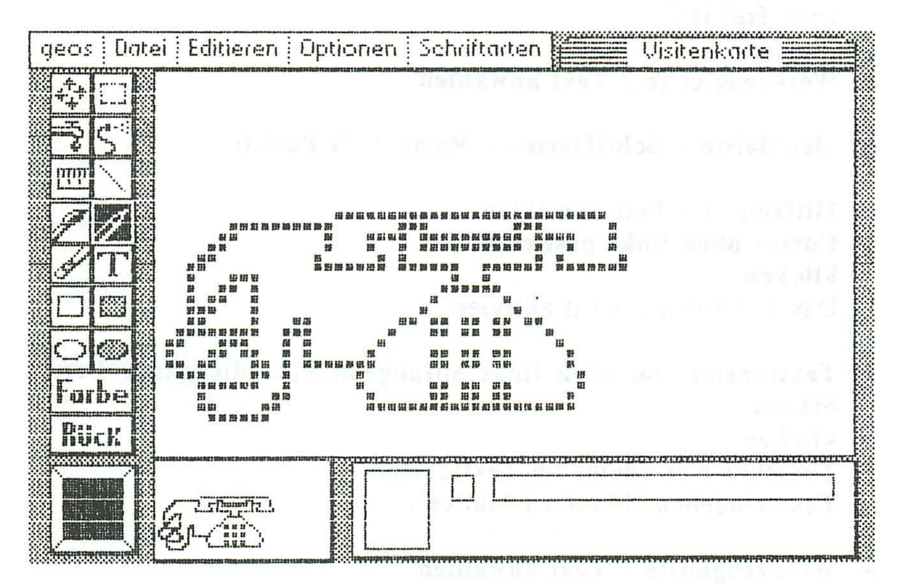
- **Werkzeugleiste / Linien anwählen**
  Zwei Linien ziehen über und unter dem Text VERSICHERUNGEN ALLER ART.

- **Werkzeugleiste / Bleistift anwählen**

- **Menüleiste / Optionen / Einzelpunkt anwählen**
  Einzelpunktrahmen erscheint oben links im Bild.

- **Den Rahmen mit dem Joystick vor die Telefonnummer bewegen.**
  klicken
  Der Inhalt des Rahmens wird im Bildausschnittfenster gezeigt.

- **Mit dem Stift ein Telefon zeichnen.**
  Hier sind der eigenen Fantasie keine Grenzen gesetzt.



- **Menüleiste / Optionen / Normalmodus anwählen**
  Bild wird wieder in Normalgröße gezeigt.

- **Werkzeugleiste / Bleistift anwählen**
  Den Bleistift mit dem Joystick ganz an den linken Rand setzen. Es muß genau sein, da aus dem Bleistift sonst wieder der Mauspfeil wird.
  klicken
  Der Bleistift wird aktiv.

- **Bewegen Sie den Stift mit dem Joystick ganz nach oben. Wenn der Stift jetzt zum Mauspfeil wird, ändert es nichts an der Funktion.**

- **Bewegen Sie den inzwischen entstandenen Mauspfeil einmal um das ganze Arbeitsblatt herum. Der Bildausschnitt wird damit komplett eingerahmt.**

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt ganz nach links verschieben, bis die erste Karte im Bild ist.
  Bildausschnitt nach unten verschieben, bis der Bildausschnitt ganz frei ist.

- **Werkzeugleiste / Text anwählen**

- **Menüleiste / Schriftarten / Roma / 24 Punkte**

- **Hilfsmenü / Fett anwählen**
  Cursor oben links postieren.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster von oben links anfangend über die ganze Breite öffnen.**
  klicken
  Textcursor erscheint im Textfenster.
  Text eingeben "Manfred Müller".

- **Werkzeugleiste / Text anwählen**

- **Menüleiste / Schriftarten / LW Cal / 10 Punkte**

- **Hilfsmenü / Normal anwählen**
  Cursor rechts neben das Feld FARBE der Werkzeugleiste bewegen.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster in waagerechter Verlängerung des Farbe Feldes der Werkzeugleiste über die ganze Breite nach unten rechts öffnen.**
  klicken
  Textcursor erscheint im Textfenster.

- **Text eingeben**
  "Mühlenstraße 13 4000 Düsseldorf Tel.0211/654321"

- **Werkzeugleiste / Pinsel anwählen**
  Mit dem Joystick den Pinsel unter das M des Vornamens bewegen.
  klicken
  Pinselfunktion ist aktiv.

- **Waagerechte Linie nach rechts ziehen.**
  Auf beiden Seiten ca. 2 Pinselbreiten freilassen.
  Eine zweite identische Linie über die Adresse ziehen.

- **Werkzeugleiste / Text anwählen**
  Cursor über der unteren Linie in Höhe des Wortes Tel. postieren.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster nach rechts über die ganze Breite und nach unten bis knapp über die Linie öffnen.**
  klicken
  Textcursor erscheint im Textfenster.
  Text eingeben "Optische Geräte".

- **Werkzeugleiste / Kreis anwählen**
  Cursor zwischen die beiden Linien in Höhe des Buchstabens A vom Vorname positionieren.
  klicken
  Kreisfunktion ist aktiv.

- **Mit dem Joystick den Kreis nach rechts öffnen, bis der Kreis fast an die Linien stößt.**
  klicken
  Der Kreis ist fixiert.

- **Den Cursor jetzt bis unter den Buchstaben N des Vornamens wieder nach links bewegen.**
  klicken
  Kreisfunktion ist aktiv.

- **Mit dem Joystick den Kreis nach rechts öffnen, bis der Kreis fast an den äußeren Kreis stößt.**
  klicken
  Der Kreis ist fixiert.

- **Cursor zwischen die beiden Linien in Höhe des Leerzeichens zwischen Vorname und Nachname positionieren.**
  klicken
  Kreisfunktion ist aktiv.

- **Mit dem Joystick den Kreis nach links öffnen, bis der Kreis fast an die Linien stößt.**
  klicken
  Der Kreis ist fixiert.

- **Den Cursor jetzt bis unter den Buchstaben D des Vornamens wieder nach rechts bewegen.**
  klicken
  Kreisfunktion ist aktiv.

- **Mit dem Joystick den Kreis nach links öffnen, bis der Kreis fast an den äußeren Kreis stößt.**
  klicken
  Der Kreis ist fixiert.

- **Werkzeugleiste / Bleistift anwählen**
  Den Bleistift mit dem Joystick ganz an den linken Rand setzen. Es muß genau sein, da aus dem Bleistift sonst wieder der Mauspfeil wird.
  klicken
  Der Bleistift wird aktiv.

- **Bewegen Sie den Stift mit dem Joystick ganz nach oben. Wenn der Stift jetzt zum Mauspfeil wird, ändert es nichts an der Funktion.**
  Bewegen Sie den inzwischen entstandenen Mauspfeil einmal um das ganze Arbeitsblatt herum. Der Bildausschnitt wird damit komplett eingerahmt.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt nach rechts verschieben, bis der Bildausschnitt ganz frei ist.

- **Werkzeugleiste / Text anwählen**

- **Menüleiste / Schriftarten / Cory / 24 Punkte anwählen**

- **Hilfsmenü / Fett anwählen**
  Den Cursor bis fast in die obere linke Ecke bewegen.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster mit dem Joystick nach rechts unten öffnen.**
  klicken
  Textcursor erscheint im Textfenster.
  Text eingeben "Manfred Müller".

- **Menüleiste / Schriftarten / Cory / 12 Punkte anwählen**

- **Hilfsmenü / Fett anwählen**
  Cursor unter das M von Manfred positionieren.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster über die ganze Breite öffnen.**
  klicken
  Textcursor erscheint im Textfenster.
  Text eingeben "Gebrauchtwagenhandel".

- **Werkzeugleiste / Text anwählen**

- **Menüleiste / Schriftarten / Cory / 12 Punkte anwählen**

- **Hilfsmenü / Normal anwählen**
  Cursor in die Bildmitte bewegen.
  klicken
  Das Textfenster wird aktiviert.

- **Textfenster bis in die rechte untere Ecke öffnen.**
  klicken
  Textcursor erscheint im Textfenster.
  Text eingeben "Mühlenstraße 13".
  RETURN drücken zum Zeilenvorschub.
  Text eingeben "4000 Düsseldorf".
  Zweimal RETURN drücken, um eine Leerzeile zu erhalten.

- **Text eingeben "0211/654321"**

- **Werkzeugleiste / Rechtecke anwählen**
  Cursor links über den Namen bewegen.
  klicken
  Rechteckfunktion ist aktiv.

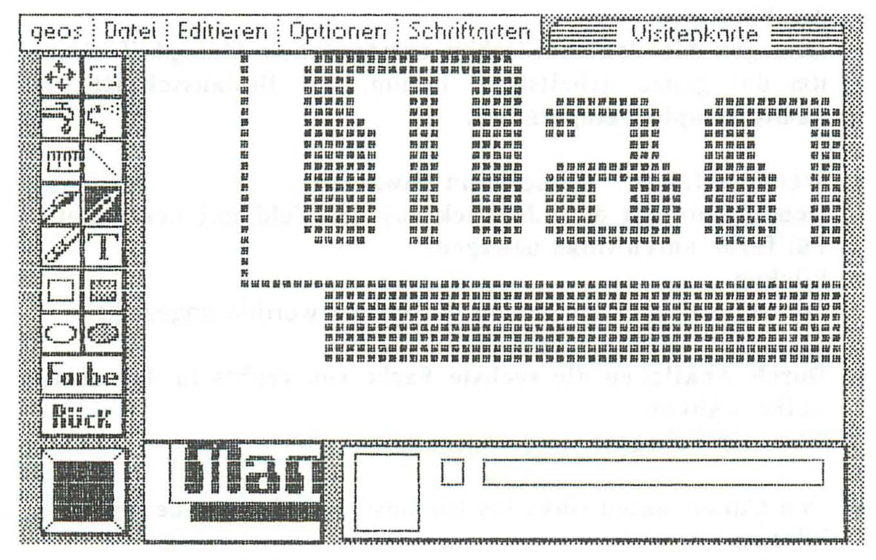
- **Mit dem Joystick das Rechteck bis rechts unter den Namen öffnen.**
  klicken
  Rechteck ist fixiert.
  Das gleiche mit den restlichen Texten.



- **Werkzeugleiste / Pinsel anwählen**

- **Menüleiste / Optionen / Pinsel wechseln**
  Den zweiten Rechteckpinsel von links wählen.
  Bewegen im Auswahlfenster mit dem Joystick, wählen durch klicken.
  Mit dem Pinsel um die Unterseite die rechte Seite der Rechtecke Linien ziehen.

- **Menüleiste / Optionen / Einzelpunkt anwählen**
  Jetzt kann die Feinarbeit gemacht werden.
  Mit dem Joystick das Auswahlfenster auf die erste Ecke bewegen.
  klicken
  Im Bildausschnittfenster wird der gewählte Bereich gezeigt.



Sollte die Ecke bei Ihnen nicht so aussehen, kann diese jetzt nachbearbeitet werden. Hierzu:

- **Werkzeugleiste / Bleistift anwählen**
  Stift ist aktiv.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Mit dem Joystick den Auswahlrahmen zur nächsten Ecke oder sonstigen Stelle, welche nachgebessert werden muß, bewegen. Wenn alles nachgebessert ist:

- **Menüleiste / Optionen / Normalmodus anwählen**

- **Werkzeugleiste / Bleistift anwählen**
  Den Bleistift mit dem Joystick ganz an den linken Rand setzen. Es muß genau sein, da aus dem Bleistift sonst wieder der Mauspfeil wird.
  klicken
  Der Bleistift wird aktiv.

- **Bewegen Sie den Stift mit dem Joystick ganz nach oben. Wenn der Stift jetzt zum Mauspfeil wird, ändert es nichts an der Funktion.**
  Bewegen Sie den inzwischen entstandenen Mauspfeil einmal um das ganze Arbeitsblatt herum. Der Bildausschnitt wird damit komplett eingerahmt.

- **Werkzeugleiste / Wasserhahn anwählen**
  Den Cursor mit dem Joystick auf das Feld mit der akuellen Füllfarbe unten links bewegen.
  klicken
  Die zur Auswahl stehenden Füllfarben werden angezeigt.

- **Durch Anklicken die sechste Farbe von rechts in der oberen Reihe wählen.**
  Aktuelle Füllfarbe wird angezeigt.

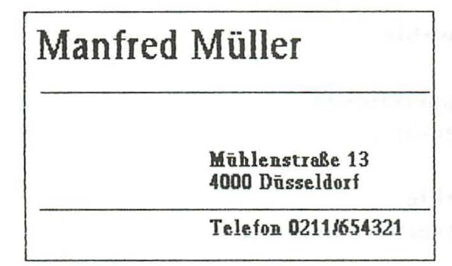
- **Den Cursor unten links ins Bildausschnittfenster bewegen.**
  klicken
  Das Feld wird mit der gewählten Farbe gefüllt.

- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnittfenster ganz nach links verschieben, bis die vorherige Karte wieder im Fenster ist.
  klicken
  Bewegungspfeile werden deaktiviert.

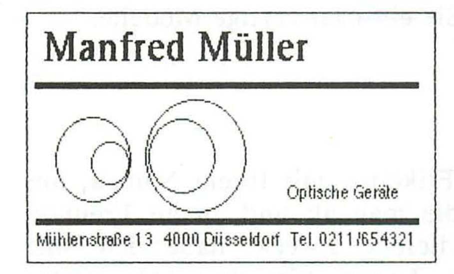
- **Werkzeugleiste / Rahmen anwählen**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Kopieren anwählen**
  Gerahmter Bildausschnitt wird zwischengespeichert.











- **Werkzeugleiste / Bewegungspfeile anwählen**
  Bildausschnitt nach unten bewegen, bis das Fenster ganz frei ist.
  klicken
  Bewegungspfeile werden deaktiviert.

- **Werkzeugleiste / Rahmen anwählen**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Menüleiste / Editieren / Einkleben anwählen**
  Der eben zwischengespeicherte Bildausschnitt wird in das markierte Feld eingeklebt.

- **Werkzeugleiste / Rahmen anwählen**

- **Werkzeugleiste / Rahmen doppelklicken**
  Ganzer Bildausschnitt wird gerahmt.

- **Hilfsmenü / Invertieren anwähle**
  Der gerahmte Bereich wird invertiert dargestellt.

In der obigen Abbildung sehen Sie ein paar fertige Modelle.

10.5 Etiketten

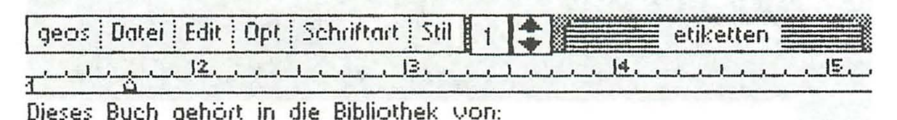
Sie benötigen sicher manchmal Etiketten mit Ihrem Namen, um z.B. Bücher zu kennzeichnen, die man ab und zu an Freunde und Bekannte ausleiht. Zu diesem Zweck habe ich mit GEOWRITE Etiketten erstellt. Laden Sie dazu als erstes GEOWRITE. Als Dokumentname tippen Sie ETIKETTEN ein. Nachdem das Programm geladen ist, haben Sie das leere Blatt auf dem Bildschirm. Der Cursor steht in der linken oberen Ecke Wir geben hier die erste Zeile Text ein.

"Dieses Buch gehört in die Bibliothek von:"

Der Text wird in der Schriftart BSW, 9 Punkt geschrieben, da diese Schriftart voreingestellt ist. Nachdem wir den Cursor durch Drücken von Return in die nächste Zeile bewegt haben, wählen wir nun eine größere Schriftart aus. In diesem Beispiel habe ich CALIFORNIA 14 Punkt gewählt. Wir geben jetzt unseren Namen und unsere Adresse ein. Sie können selbstverständlich Ihren eigenen Namen nehmen:

```
"Max Maier"
"Mühlerstraße 23"
"4000 Düsseldorf 1"
"Telefon 0211/654321"
```

Jetzt müßte es auf Ihrem Bildschirm eigentlich so aussehen, natürlich mit Ihrem Namen.

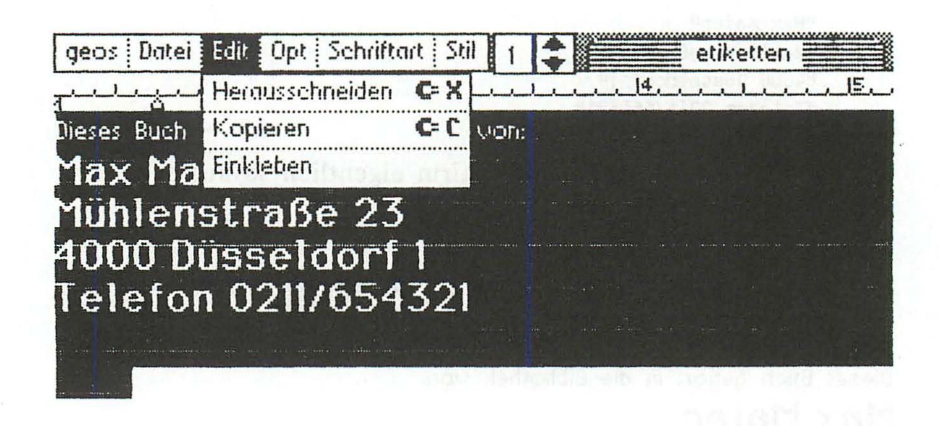


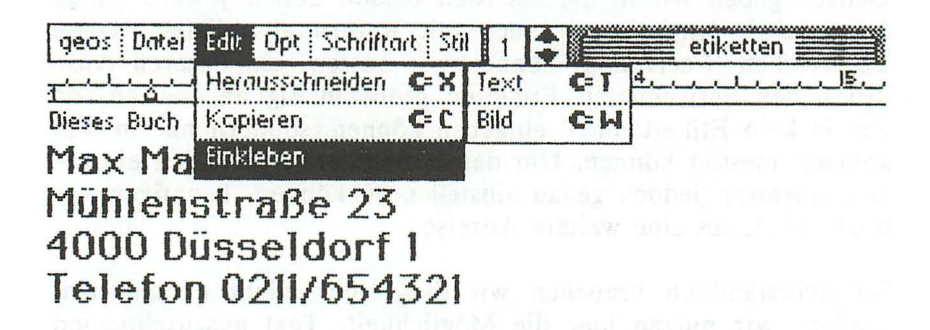
Max Maier
Mühlenstraße 23
4000 Düsseldorf 1
Telefon 0211/654321

Danach geben wir in die nächsten beiden Zeilen jeweils einige Leerzeichen ein, die wir auch mit Return abschließen. Diese Leerzeichen (Leerzeilen) können wir nutzen, um unseren Ausdruck den verwendeten Etiketten größenmäßig anzupassen, da wir ja kein Etikettenmaß eingeben können, sondern nur in 'Pixelmaß' messen können. Um den Abstand zwischen den einzelnen Adressen jedoch genau einstellen zu können, benötigen wir noch mindestes eine weitere Adresse.

Selbstverständlich brauchen wir diese jetzt nicht einzutippen, sondern wir nutzen hier die Möglichkeit, Text auszuschneiden und zwischenzuspeichern. Markieren Sie hierzu bitte den gesamten Text incl. der beiden Leerzeilen, und wählen Sie unter der Option EDIT die Funktion KOPIEREN.

Nachdem der markierte Text gespeichert ist, bewegen Sie den Cursor unter die 2. Leerzeile an den Zeilenanfang und wählen nun wiederum unter der Option EDIT den Punkt EINKLEBEN und dann TEXT. Sie sehen, der eben gespeicherte Text wird von der Cursorposition an auf den Bildschirm ausgegeben.





Damit ist der Text jedoch nicht verschwunden, sondern er bleibt solange zwischengespeichert, bis ein neuer Text ausgeschnitten oder kopiert wird. Bewegen Sie den Cursor also nochmals an das Textende unter die beiden Leerzeilen, und wählen Sie TEXT EINKLEBEN. Es erscheint wieder die gespeicherte Adresse.

Jetzt sollten Sie den ersten Probeausdruck machen. Da Etiketten einiges mehr kosten als normales Papier, machen Sie Ihre Probeausdrucke ruhig auf Endlospapier und halten es neben Ihre Etiketten, um zu prüfen, ob die Adressen den richtigen Abstand voneinander haben. Sollte der Abstand nicht OK sein, haben Sie die Möglichkeit, diesen zu vergrößern oder zu verkleinern, indem Sie die Leerzeilen markieren und eine andere Schriftart oder -größe wählen. Sobald der Abstand Ihren Etiketten entspricht, kleben Sie noch so viele Etiketten untereinander, bis die erste Seite voll ist. Sie können jetzt Etiketten für all Ihre Bücher drucken. Sie können den Text aber auch erst mal speichern und wann immer Sie wollen, direkt vom Desktop aus Ihre Etiketten ausdrucken.

Dieses Buch gehört in die Bibliothek von:

Max Maier
Mühlenstraße 23
4000 Düsseldorf 1
Telefon 0211/654321

10.6 Einrichtung

Diese Anwendung mit Geopaint ist eigentlich noch recht jung. Geboren wurde die Idee dadurch, daß wir unsere Küchenmöbel in einer neuen Wohnung unterbringen wollten und ich weder die Möbel gerne zigmal hin- und herschieben wollte und auch das Ausschneiden der einzelnen Schränke in der richtigen Plangröße, um diese dann über den Wohnungsplan zu schieben, mir nicht als ideale Möglichkeit erschien. Also habe ich GEOS geladen und probiert, wie so etwas mit Geopaint funktioniert. Doch wir wollen dies ja jetzt zusammen versuchen.

Laden Sie also GEOPAINT und geben "KÜCHE" als Dokumentname ein. Wählen Sie die Linienfunktion. In dem Hilfsmenü unten rechts sehen Sie ein X: und ein Y:. Hier werden beim Linien zeichnen die Punkte mitgezählt. Da der Raum 2,32 m breit und 3,48 m lang ist, zeichnen wir uns ein Rechteck, bei wel-

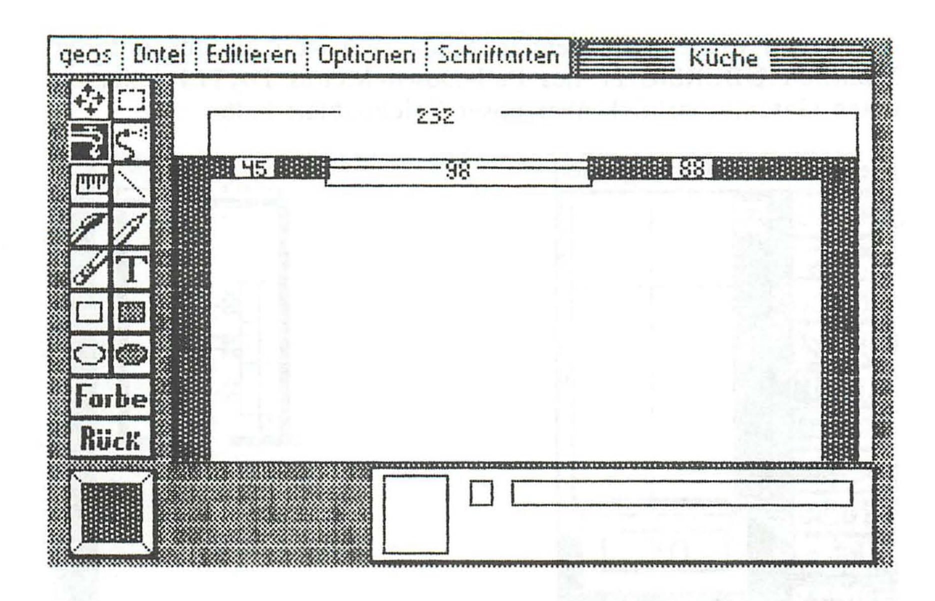
chem die waagerechte Linien 232 Punkte und die senkrechten Linien 348 Punkte lang sind. Außerdem zeichnen wir natürlich ein, von wo bis wo sich das Fenster geht und die Tür befindet.

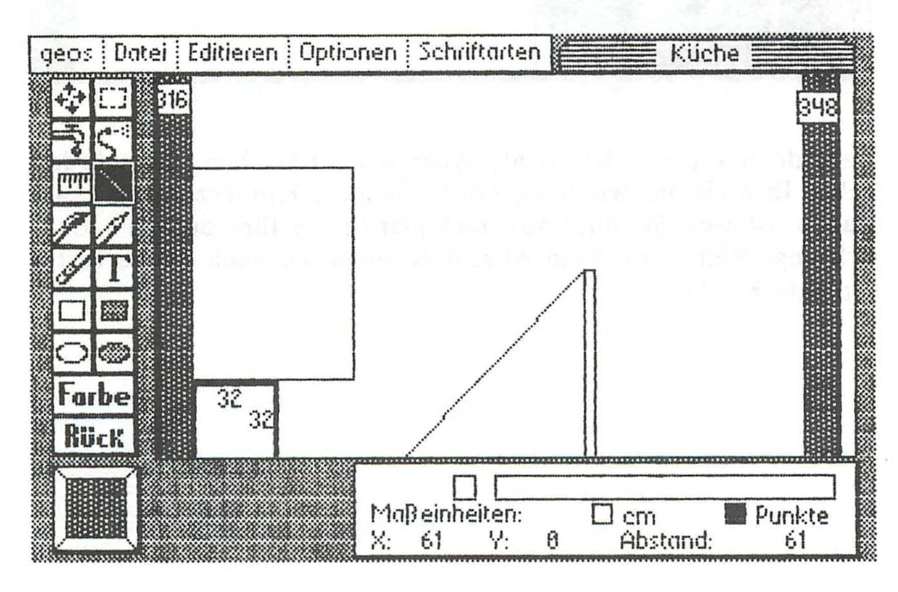
Da zumindest die senkrechten Linien nicht in einem Stück zu zeichnen sind, zeichnen wir diese in drei Etappen. Beim ersten Mal sollte die Linie 120 Punkte lang sein, dann sollten Sie das Bildausschnittfenster runterscrollen und nochmal eine Linie zeichenen, die 120 Punkte lang ist. Scrollen Sie nun noch einmal runter und zeichnen Sie nun eine Linie, die 108 Punkte lang ist. Schliessen Sie nun das Rechteck mit der unteren waagerechten Linien. Bitte vergessen Sie die Tür nicht.

Da wir sehen wollen, daß es sich um Mauern handelt, ziehen wir um unser Rechteck ein zweites, welches etwas größer ist. Die Zwischenräume (bitte darauf achten, daß die "Mauern" rundum geschlossen sind) können wir mit der Funktion FÜLLMODUS mit einem beliebigen Muster füllen. Geben Sie jetzt noch die Punkt- (cm) Maße in die Mauern ein, damit Sie immer wissen, wieviel Platz Sie zur Verfügung haben. So müßte es auch bei Ihnen jetzt eigentlich aussehen:

Da wir nicht auf der oberen (Fenster-) Seite, sondern auf der unteren (Tür-) Seite mit der Einrichtung anfangen wollen, scrollen wir den Bildausschnitt soweit runter, bis wir die untere Mauer sehen. Jetzt können wir anfangen, unsere Möbel einzuzeichnen. Da der Wasseranschluß auf der linken Seite am Anfang liegt, zeichnen wir zuerst die Spüle (80 cm breit, 60 cm tief) ein.Wir nehmen hierfür wieder die Funktion LINIEN und zeichnen für jeden cm einen Punkt.

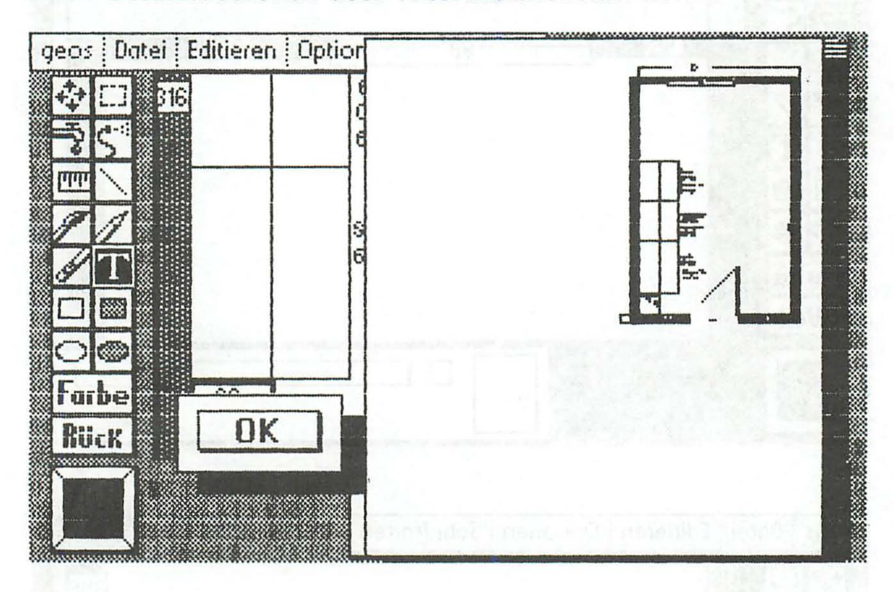
Die Oberschränke müssen natürlich auch eingezeichnet werden. Der erste Oberschrank ist 80 cm breit, 30 cm tief. Sollten Sie Probleme haben, den Pfeil rechtzeitig zu stoppen, wählen Sie unter der Option GEOS den Punkt VOREINSTELLUNG an und stellen die Anfangs- und Endgeschwindigkeit etwas zurück. Als nächstes haben wir die Spülmaschine und den Herd, beides je 60 cm breit und 60 cm tief.



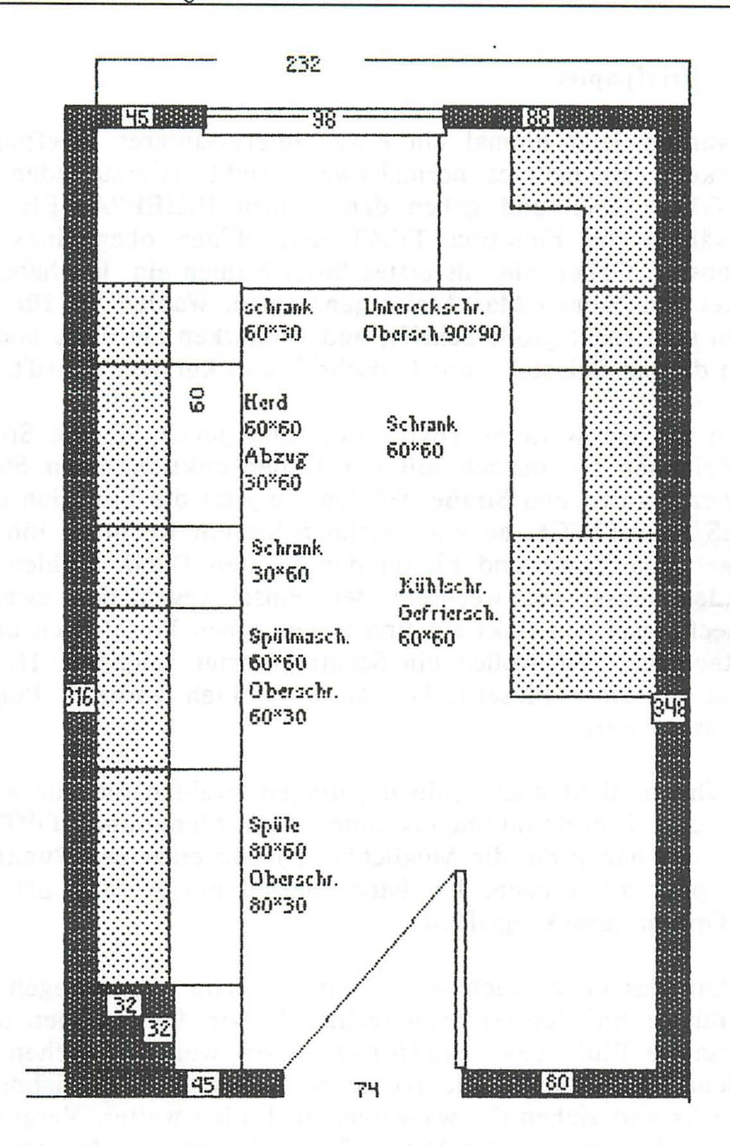


Die Oberschränke sind je 60 cm breit und 30 cm tief. Damit wir wissen, welche Möbel wir schon untergebracht haben, öffnen wir jeweils rechts neben den Möbeln ein Textfenster und schreiben in einer kleinen Schrift (Roma 9), welches Möbelstück

es ist und die entsprechenden Maße. Wir können uns mit der Funktion ÜBERSICHT des Pull-Down-Menüs DATEI jederzeit einen Gesamtüberblick über unser Zeichenblatt holen.



Geht doch super, oder nicht? Aber ich sehe schon, Sie würden lieber Ihre eigene Wohnung (Wohnzimmer, Kinderzimmer usw.) planen. Messen Sie alles aus, und planen Sie Ihre optimale Einrichtung. Viel Spaß! Zum Abschluß sehen Sie noch meine fertig geplante Küche.



Es hat alles beim ersten Mal gepaßt.

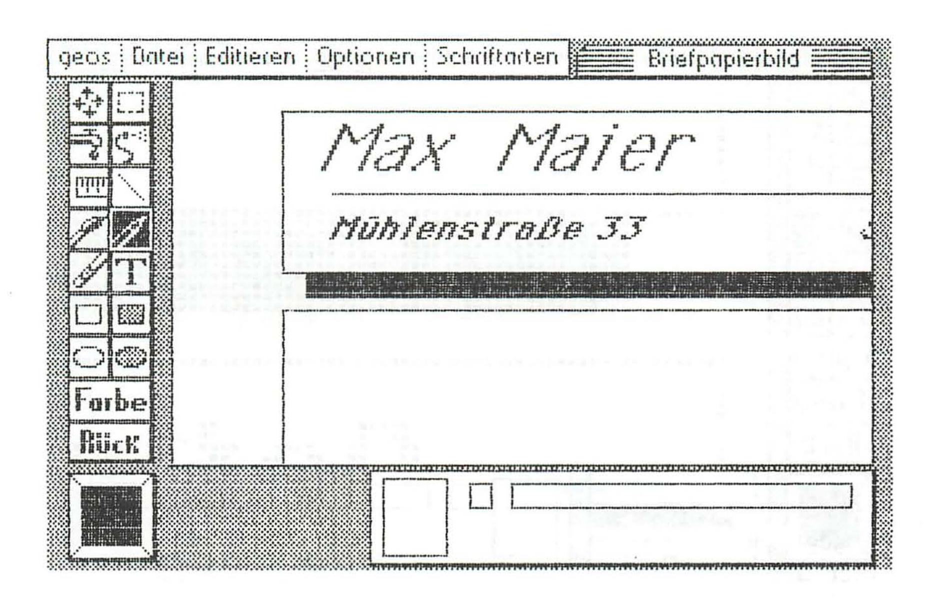
10.7 Briefpapier

Wir werden jetzt einmal ein etwas interessanteres Briefpapier entwickeln, als man es normalerweise sieht. Hierzu laden Sie bitte GEOPAINT und geben den Namen BRIEFPAPIER ein. Wir wählen die Funktion TEXT und öffnen oben links ein Textfenster. Geben hier als erstes Ihren Namen ein. Ich habe als Beispiel den Namen Max Maier genommen. Wählen Sie für den Namen eine recht große Schrift, und verstärken Sie diese noch - wie in diesem Beispiel - mit Fettschrift und kursiver Schrift.

Öffnen Sie ein weiteres Textfenster, und geben Sie die Straße ein. Zeichnen Sie danach mit der Linienfunktion einen Strich zwischen Namen und Straße. Wählen Sie jetzt die Funktion OFFENES RECHTECK an und zeichnen Sie ein Rechteck um die eingegebenen Texte und bis an den rechten Rand. Wählen Sie jetzt den Pinsel an, wechseln den Pinsel gegen den zweiten Rechteckpinsel von links aus und ziehen einen Pinselstrich unter das Rechteck. Wir wollen ein Schattenfenster zeichnen. Hierzu müssen bei einer Pinselstärke von 8 Punkten vorne 8 Punkte freigelassen werden.

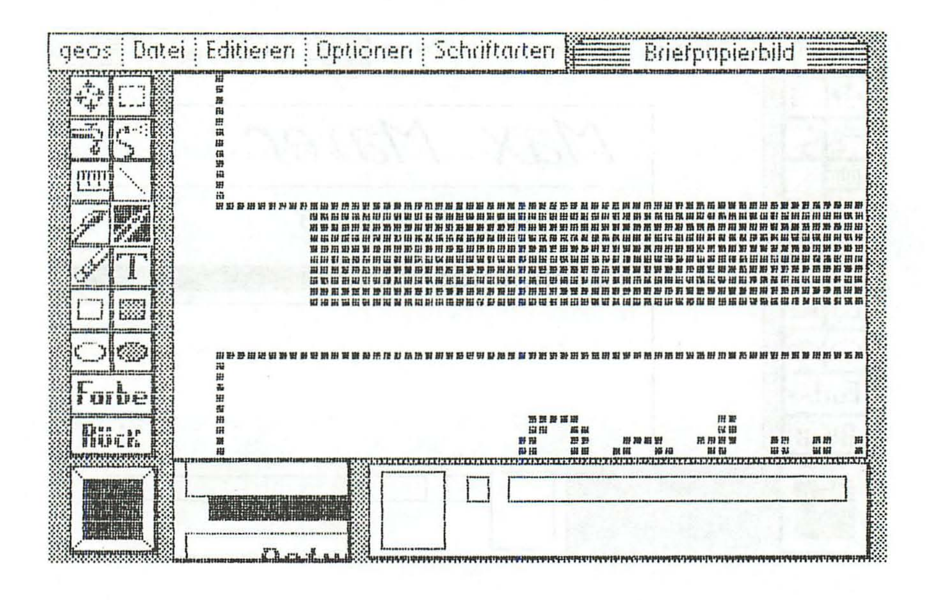
Sollte Ihnen nicht alles optimal gelingen, wählen Sie zur Korrektur den Einzelpunktmodus unter dem Menüpunkt OPTIONEN. Hier haben Sie die Möglichkeit, jeden einzelnen Punkt zu setzen oder zu löschen. Ich habe speziell am Anfang oft auf diese Option zurückgegriffen.

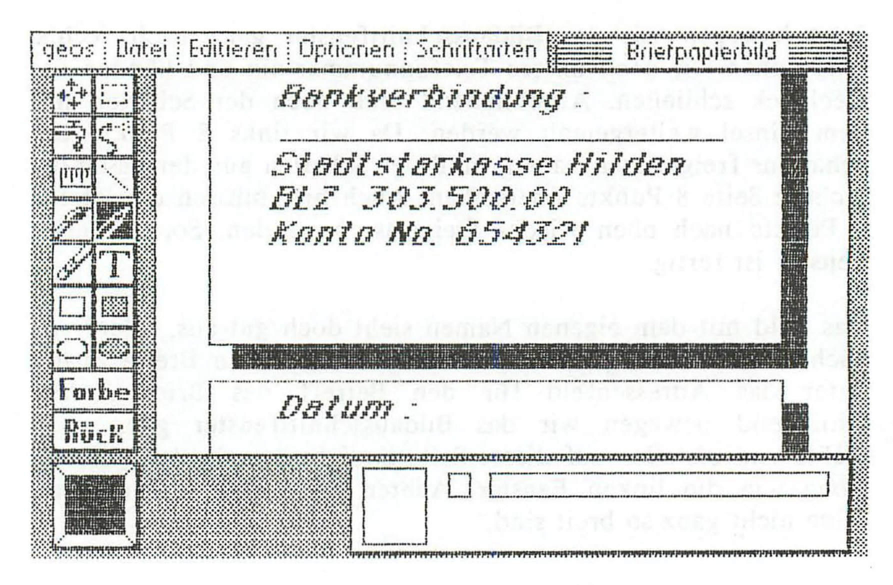
Nachdem das erste Stück des Schattens fertig ist, bewegen Sie das Bildausschnittfenster nach rechts, da wir den Rahmen über die gesamte Bild- bzw. Briefbreite ziehen wollen. Löschen Sie mit dem Radiergummi die rechte Seite des eben gezeichneten Rechtecks und ziehen die waagerechten Linien weiter. Vergessen Sie nicht die Linie in der Mitte. Geben Sie jetzt in der unteren Hälfte den Wohnort ein, so daß er auf gleicher Höhe wie links die Straße ist, und ziehen unten den Schattenrahmen mit dem Pinsel weiter.



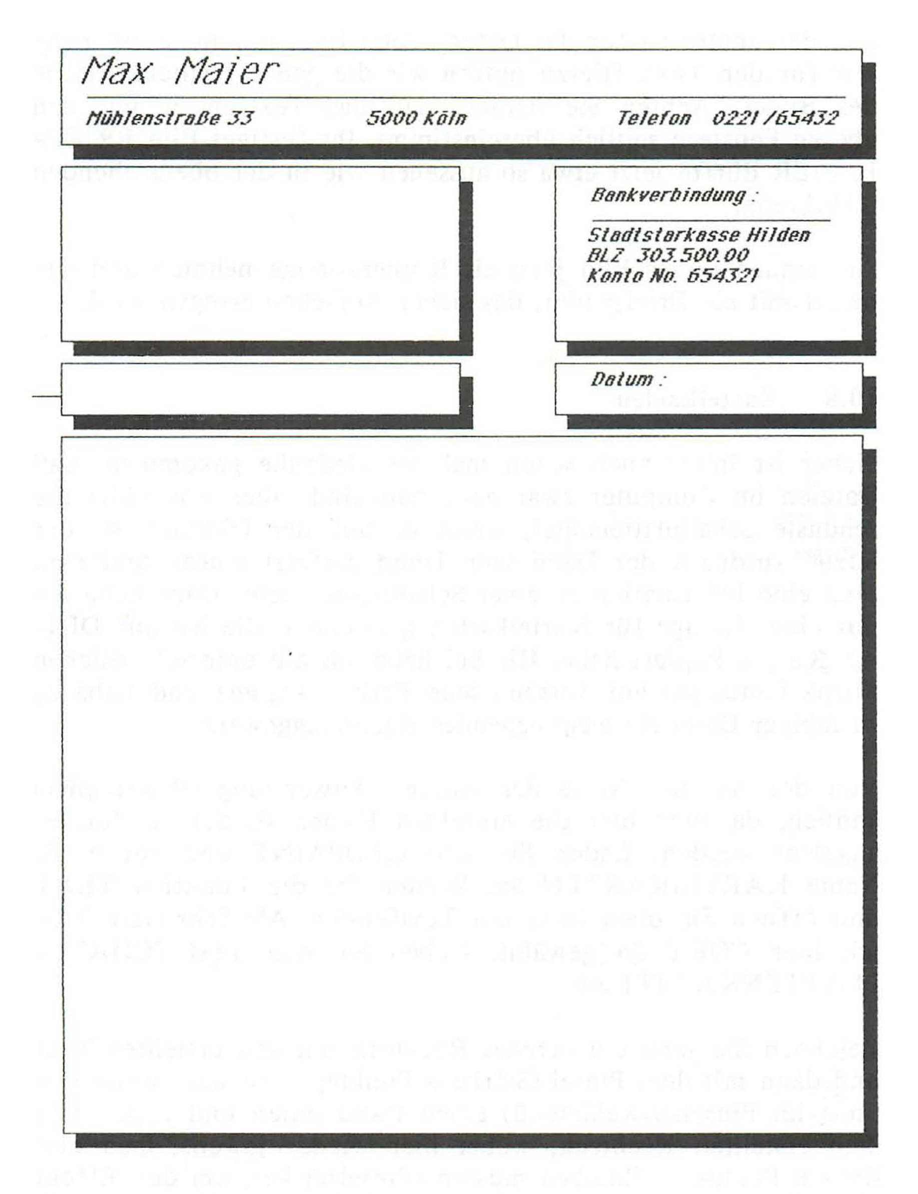
Jetzt bewegen wir das Bildausschnittfenster ganz nach rechts. Hier geben wir noch unsere Telefonnummer ein und können das Rechteck schließen. Anschließend muß noch der Schatten mit dem Pinsel weitergemalt werden. Da wir links 8 Punkte des Schattens freigelassen haben, muß der Schatten auf der gesamten rechten Seite 8 Punkte überstehen. Auch hier müssen die letzten 8 Punkte nach oben wieder freigelassen werden. So, das erste Fenster ist fertig.

Das Feld mit dem eigenen Namen sieht doch gut aus, oder? Als nächstes zeichnen wir ein Fenster in der gleichen Breite knapp unter das Adressenfeld für den Betreff des Briefes. Anschließend bewegen wir das Bildausschnittfenster ganz nach rechts und erstellen auf dieser Seite zwei Fenster in der gleichen Höhe wie die linken Fenster. Achten Sie jedoch darauf, daß diese nicht ganz so breit sind.





Bewegen Sie nun den Bildausschnitt auf die linke Seite, da wir als nächstes das Adressenfeld erstellen wollen. Achten Sie darauf, daß die Rechtecke untereinanderstehen, und zeichnen Sie ein Rechteck in der Größe des Bildausschnittes.



Hier sollten Sie einen Probeausdruck machen, um zu prüfen, ob das Adressenfeld in das Klarsichtfeld eines Briefumschlags paßt. Paßt es noch nicht, müssen Sie das Rechteck entsprechend versetzen. In das obere Fenster geben wir unsere Bankverbindung

ein, das untere ist für das Datum. Jetzt fehlt nur noch ein Fenster für den Text. Hierzu nutzen wir die ganze restliche Fläche des Bildes. Achten Sie darauf, daß das Textfenster mit den oberen Fenstern seitlich übereinstimmt. Ihr fertiges Bild BRIEF-PAPIER dürfte jetzt etwa so aussehen wie in der obenstehenden Abbildung.

Sie können dieses Bild jetzt als Kopiervorlage nehmen und haben damit ein Briefpapier, das sicher Aufsehen erregen wird.

10.8 Karteikarten

Sicher ist Ihnen auch schon mal der Gedanke gekommen, daß Dateien im Computer zwar ganz nett sind, aber was nützt die schönste Schallplattendatei, wenn sie auf der Diskette ist, der letzte Ausdruck der Datei vom Hund zerfetzt wurde und man jetzt eine Information zu einer Schallplatte sucht. Dazu habe ich mir eine Vorlage für Karteikarten gezeichnet, die ich auf DIN-A5-Karton kopiert habe. Hierbei habe ich die unterschiedlichen Musikrichtungen auf verschiedene Farben kopiert und habe so zu meiner Datei ein ansprechendes Nachschlagewerk.

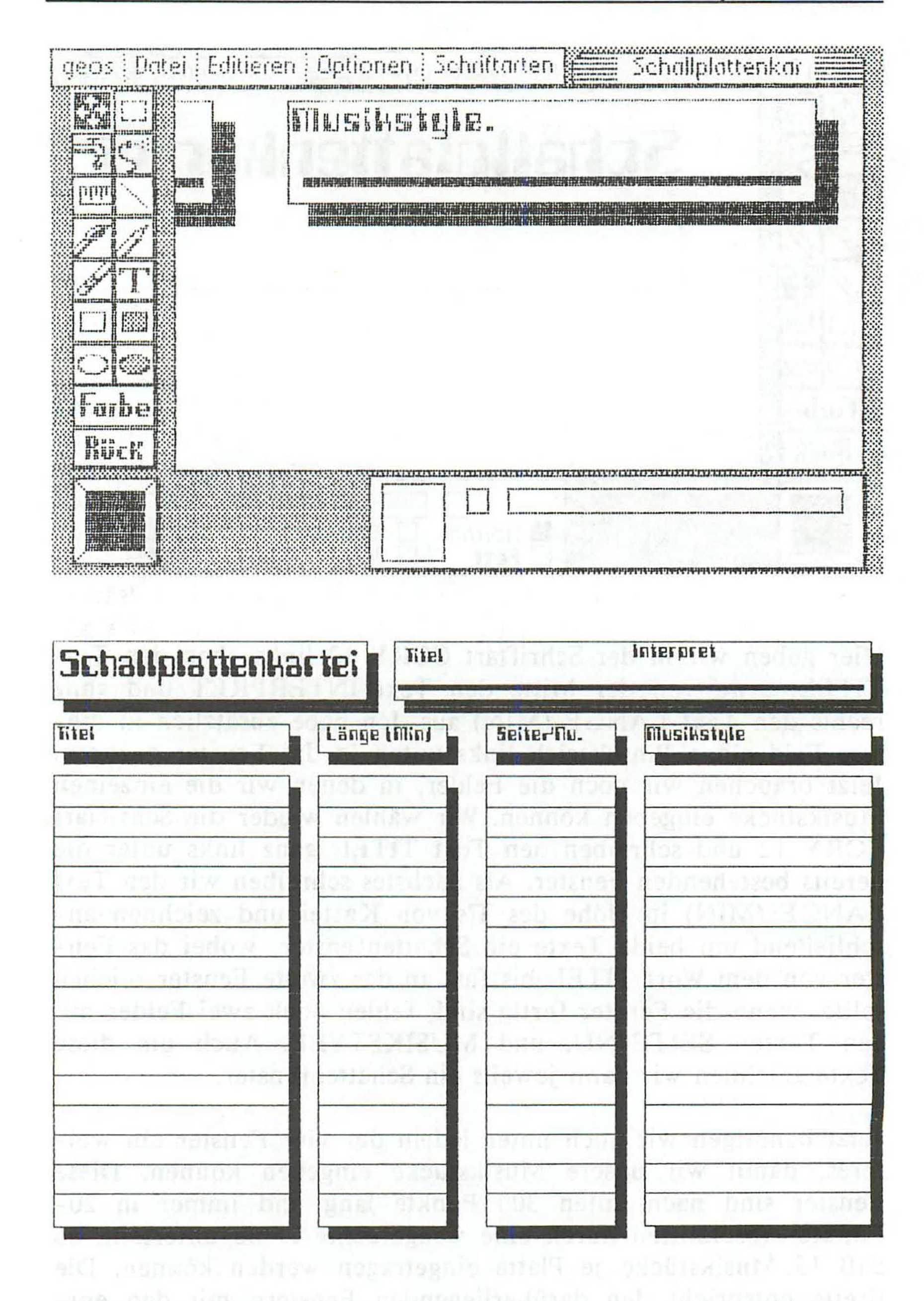
Von der Art her ist es der vorigen Anwendung (Briefpapier) ähnlich, da auch hier die einzelnen Felder als Schattenfenster angelegt werden. Laden Sie also GEOPAINT und geben als Name KARTEIKARTEN an. Wählen Sie die Funktion TEXT und öffnen Sie oben links ein Textfenster. Als Schriftart habe ich hier CORY 24 gewählt. Geben Sie den Text SCHALL-PLATTENKARTEI ein.

Zeichnen Sie jetzt ein offenes Rechteck um den erstellten Text und dann mit dem Pinsel (Stärke 8 Punkte; es ist der zweite von links im Pinselauswahlmenü) einen Rand unten und rechts von dem erstellten Rechteck, wobei hier wieder jeweils oben und links 8 Punkte freibleiben müssen (Pinselstärke), um den Effekt eines Schattens zu erhalten. Ein identisches Fenster zeichnen wir rechts davon bis zum rechten Rand.



Hier geben wir in der Schriftart CORY 12 links oben den Text TITEL, etwa von der Mitte den Text INTERPRET und ganz rechts den Text LÄNGE (MIN) aus. Ich habe zusätzlich in diesem Feld einen Pinselstrich links unten in das Fenster gezogen. Jetzt brauchen wir noch die Felder, in denen wir die einzelnen Musikstücke eingeben können. Wir wählen wieder die Schriftart CORY 12 und schreiben den Text TITEL ganz links unter die bereits bestehenden Fenster. Als nächstes schreiben wir den Text LÄNGE (MIN) in Höhe des T's von Kartei und zeichnen anschließend um beide Texte ein Schattenfenster, wobei das Fenster von dem Wort TITEL bis fast an das zweite Fenster reichen sollte. Wenn die Fenster fertig sind, fehlen noch zwei Felder mit den Texten SEITE/NU. und MUSIKSTYLE. Auch um diese Texte zeichnen wir dann jeweils ein Schattenfenster.

Jetzt benötigen wir noch unter jedem der vier Fenster ein weiteres, damit wir unsere Musikstücke eingeben können. Diese Fenster sind nach unten 300 Punkte lang und immer in 20-Punkte-Abschnitten durch eine waagerechte Linie unterteilt, so daß 15 Musikstücke je Platte eingetragen werden können. Die Breite entspricht den darüberliegenden Fenstern mit den entsprechenden Überschriften.



Sollten Sie noch irgendeine Angabe zusätzlich aufnehmen wollen, können Sie natürlich die Karteikarte nach Ihren eigenen

Wünschen erweitern oder ändern. Vielleicht erstellen Sie auf diese Art Ihre Karteikarten für Kochrezepte. Es gibt hier sicherlich eine ganze Menge Möglichkeiten, die diese Art der Darstellung zu nutzen.

10.9 Einladung

Wir wollen jetzt einmal eine Einladungskarte erstellen, wobei ich gestehen muß, daß diese Idee nicht von mir stammt, sondern von einem anderen Programm übernommen wurde. Mich hat bei der Grundversion der Karte immer geärgert, daß ich bei Texteingaben eingeschränkt war. So ist diese GEOPAINT-Anwendung geboren worden. Ich möchte hier keine Schriftart vorgeben. Wählen Sie bitte die Schrift, die Ihnen am besten gefällt. Als erstes zeichnen wir einen Rahmen um das gesamte Blatt. Bevor wir jetzt anfangen, muß ich doch noch etwas Grundsätzliches zu dieser Anwendungsart sagen, da sicher der eine oder andere unter Ihnen nicht den vorgegebenen Rahmen zeichnen wollen, sondern hier vielleicht gleich eine eigene Idee einfließen lassen wollen.

Beim Zeichen von immer wiederkehrenden Motiven, wie z.B. der gleich beschriebene Rahmen, sollte man grundsätzlich immer in 8er Schritten zeichnen. Ich habe bei meinen ersten Zeichnungen dieser Art einfach drauflos gezeichnet und habe mich fürchterlich darüber geärgert, daß die Funktionen VERSCHIEBEN, DREHEN, KOPIEREN usw. mir oftmals große Probleme bereitet haben, weil ich ein anderes Grundmaß benutzt habe. Das liegt ganz einfach daran, daß u.a. die genannten Funktionen in 8er-Schritten arbeiten. Sicher ist Ihnen auch schon aufgefallen, daß beim Verschieben von einem markierten Grafikteil dieser Teil meistens nicht an der Stelle abgesetzt wird, wo man abklickt, sondern daß er dann noch ein Stück versetzt wird. GEOPAINT paßt den markierten Bereich in diesem Moment in dieses 8*8-Grundmaß ein. Damit aber genug der grauen Theorie.

Wir beginnen mit unserem Rahmen in der linken oberen Ecke. Wählen Sie hierzu den Einzelpunktmodus aus, um unser Grundmotiv zu zeichnen. Wir wollen einen doppelreihiges Quadrat von

16 * 16 Punkten erstellen und in der Mitte ein Quadrat von 4 * 4 Punkten einsetzen. Hier haben wir übrigens gleich wieder eine Sache, auf die dringend geachtet werden muß. Die oberen wie auch die unteren zwei Punktreihen des Bildes sind im Einzelpunktmodus nicht erreichbar. Wir denken uns also die oberen zwei Punktreihen als gezeichnet, fangen mit der dritten Reihe an und zeichnen links und rechts je eine Doppelreihe Punkte, also jeweils 14 Punkte senkrecht, und zeichnen dann die untere Doppelreihe (16 Punkte waagerecht).

Anschließend zeichnen wir das Quadrat (4* 4 Punkte) in die Mitte ein, so daß wir einen Abstand von 4 Punkten vom inneren Quadrat nach außen haben. Wir setzen das gleiche direkt noch einmal darunter. Jetzt muß die obere waagerechte Doppelreihe aber mitgezeichnet werden. Wenn Sie fertig sind, schalten Sie wieder in den Normalmodus zurück. Sicher wissen Sie ja, daß Sie den Einzel- und Normalmodus außer über die Menüleiste auch durch Doppelklicken des gewählten Werkzeugs BLEISTIFT erreichen können.

Wählen Sie jetzt die Funktion MARKIEREN aus der Werkzeugleiste aus, und markieren Sie das untere der beiden Grundelemente des Rahmens. Beim Markieren halten Sie den Mauspfeil innerhalb des zu markierenden Bereichs, denn hier schlägt das 8*8-Grundmaß durch, und Sie sehen, daß der Markierungsrahmen sich genau um unser Grundelement legt, obwohl Sie nicht genau auf den Eckpunkten geklickt haben. Um erstmal die oberen beiden noch fehlenden Punktreihen einzufügen, wählen Sie bitte KOPIEREN an und bewegen den markierten Teil in die linke obere Ecke. Damit ist auch das obere Element fertig. Jetzt markieren Sie beide Elemente und kopieren diese unter die beiden ersten. Da nun vier Grundelemente vorhanden sind, markieren wir diese und kopieren abermals. Sie sehen, es geht doch recht schnell.

Nachdem der senkrechte Rahmen (9 Elemente) soweit fertig ist, markieren Sie alle vorhandenen Elemente und wählen die Funktion DREHEN an. Es erscheint eine Kopie des markierten Bereiches um 90 Grad gedreht. Diese verschieben wir jetzt in die obere linke Ecke. Sie haben jetzt also senkrechte und waage-

rechte Grundelemente, die Sie jetzt nur noch um das ganze Bild kopieren müssen.

Da hierbei der Bildausschnitt verschoben werden muß, nutzen Sie die Editierfunktion der Menüleiste. Markieren Sie z.B. nur die waagerechen Elemente, und kopieren Sie diese mit der Editierfunktion. Dann verschieben Sie den Bildausschnitt nach rechts, markieren den Bereich, in den die nächsten Rahmenelemente eingeklebt werden sollen und wählen unter EDITIEREN die entsprechende Funktion. Sicher haben Sie jetzt erstmal eine Zeitlang zu tun. Lassen Sie für diese Anwendung bitte die unteren 16 Punktreihen frei. Sie setzen also 40 Elemente nebeneinander und 44 Elemente untereinander. Nachdem Sie den Rahmen fertig haben, müssen jetzt noch zwei senkrechte und zwei waagerechte Reihen mit den Rahmenelementen einkopiert werden, und zwar so, daß wir anschließend vier gleichgroße Felder haben.

Jetzt müssen noch die Texte in die einzelnen Felder eingefügt werden. Da wir eine Einladungskarte zeichnen wollen, benötigen wir folgende Textfelder: Im Feld oben links zwei Textfelder, je eins im oberen und eins im unteren Bereich des Feldes. Geben Sie in den oberen Bereich den Text für

```
Hans und Gerda Meier
```

und im unteren Bereich den Text

```
Einladung
```

ein. Markieren Sie die einzelnen Textfenster, und spiegeln Sie diese jetzt erst um die X-Achse und anschließend um die Y-Achse. Sie sehen, der Text steht jetzt auf dem Kopf. Im Feld oben rechts geben Sie nur im oberen Bereich einen Text ein:

```
Bitte telefonisch bestätigen 0211/654321
```

Auch dieser Text wird um die X- und Y-Achse gespiegelt. Im Feld unten links brauchen wir wieder zwei Textfenster. In das obere Textfeld geben Sie ein:

```
zur 35. Geburtstagsfeier
```

Und in das untere Textfeld:

```
von
Max Maier
```

Im Feld rechts unten erstellen wir drei Textfenster

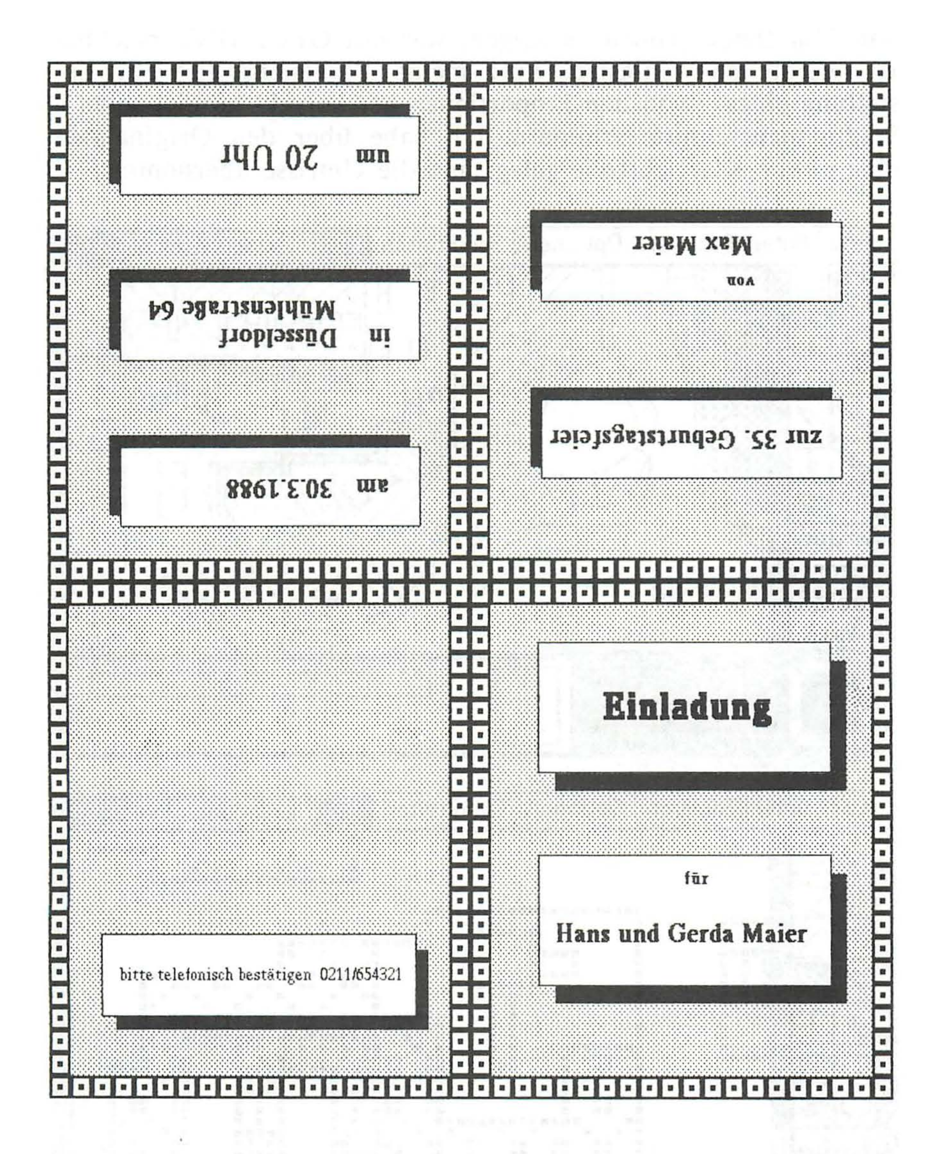
```
am 30. 3.1988
in Düsseldorf
Mühlenstraße 64
um 20 Uhr
```

Jetzt habe ich die Textfenster gerahmt und wie in den beiden vorangegangenen Beispielen zu Schattenfenstern gemacht. Sollten Sie auch Schattenfenster machen wollen, denken Sie daran, daß in den oberen beiden Fenstern der Schatten seitenverkehrt angebracht werden muß, (anstatt unten und rechts hier oben und links).

Warum die beiden oberen Felder auf dem Kopf stehen, werden Sie gleich sehen. Lassen Sie sich hierzu die Einladung einmal ausdrucken. Schneiden Sie bitte die weißen Ränder mit einer Schere ab, knicken Sie die Einladung einmal an der waagerechten Mittellinie und anschließend an der senkrechten Mittellinie ein.

Sie müßten jetzt eine Einladungskarte in der Hand haben, auf der vorne EINLADUNG steht, hinten die Bitte um telefonische Bestätigung und nach dem einmaligen Aufklappen haben Sie links den Grund der Einladung, und rechts, wo und wann Sie eingeladen sind.

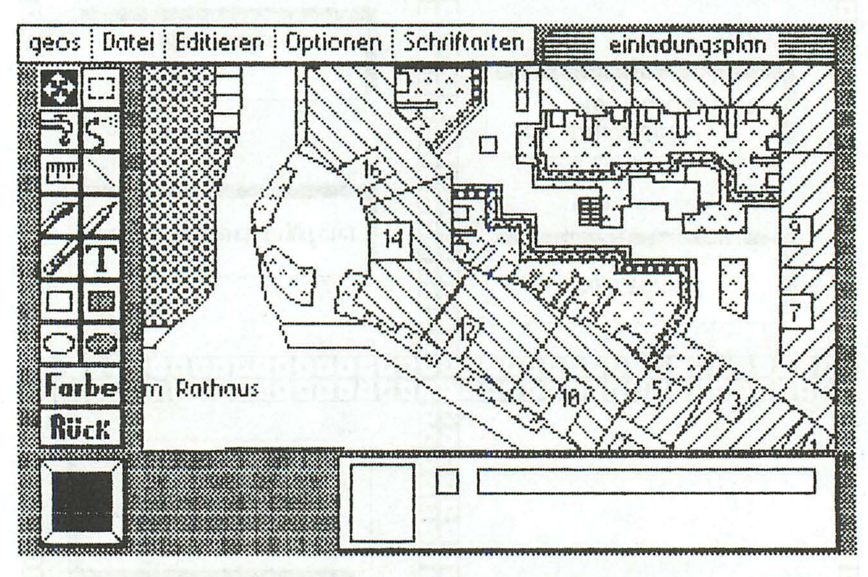
Vielleicht haben die Texte noch nicht die richtige Form, aber Sie können die Textfenster ja noch verschieben bzw. Texte nach eigener Wahl oder nur Schattenfenster ohne Text anlegen. Ebenso wie die Karteikarte können Sie natürlich auch diese Karte als Kopiervorlage nutzen, die Einladung doppelseitig auf DIN-A5-Karton kopieren und die leeren Schattenfenster handschriftlich ausfüllen. Hier nun die fertige Einladung:

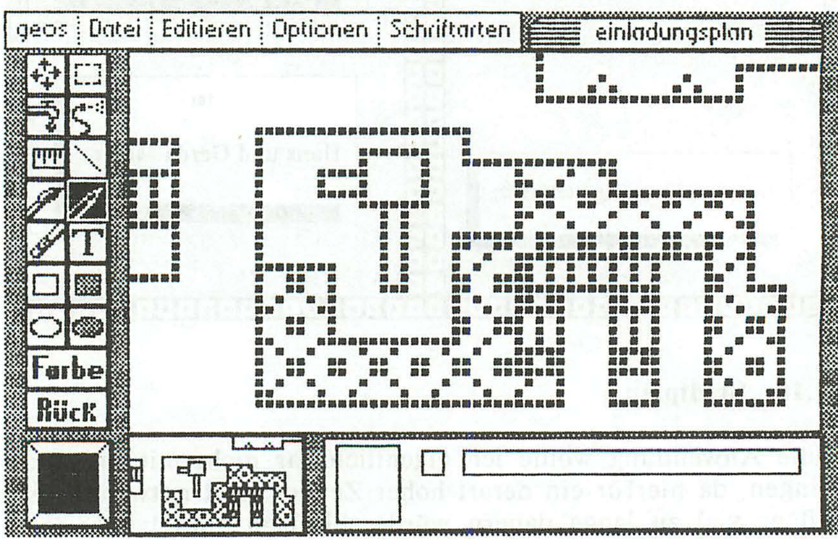


10.10 Stadtplan

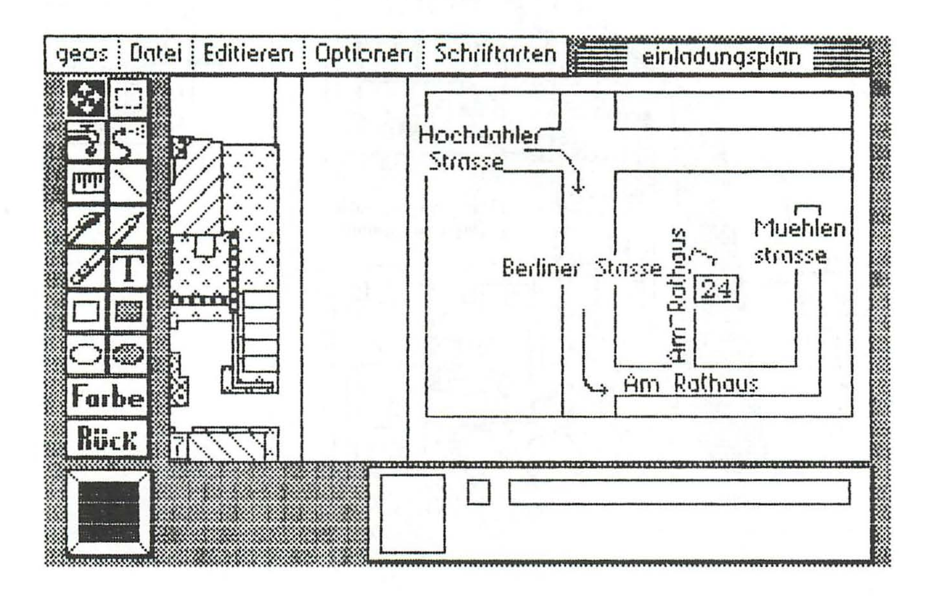
Diese Anwendung wollte ich eigentlich gar nicht mit ins Buch bringen, da hierfür ein derart hoher Zeitaufwand notwendig ist, daß es viel zu lange dauern würde, bis man alles beschrieben

hat. Um Ihnen jedoch zu zeigen, was mit GEOPAINT machbar ist, werde ich wenigstens kurz meine Vorgehensweise zur Erstellung eines - von den Proportionen her korrekt gezeichneten - Stadtplanausschnitts schildern. Ich habe über den Originalplan eine dünne Klarsichtfolie gelegt und die Umrisse übernommen.

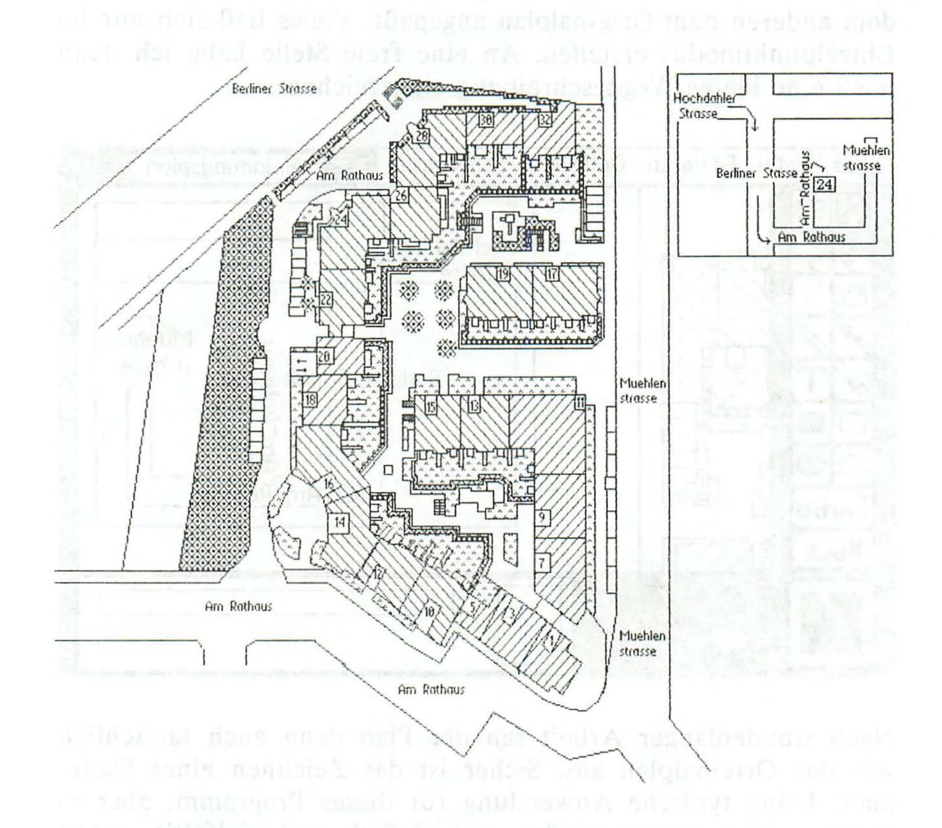




Diese Folie habe ich dann auf den Bildschirm gelegt (wenn die Folie dünn genug ist, haftet diese sehr gut) und habe die Umrisse mit GEOPAINT gezeichnet. Die Folie habe ich mit dem Verschieben des Bildausschnittes identisch mitverschoben. Nachdem die Umrisse fertig waren, wurde ein Bildausschnitt nach dem anderen dem Originalplan angepaßt. Vieles ließ sich nur im Einzelpunktmodus erstellen. An eine freie Stelle habe ich dann noch eine kleine Wegbeschreibung eingezeichnet.



Nach stundenlanger Arbeit sah der Plan dann auch tatsächlich wie der Originalplan aus. Sicher ist das Zeichnen eines Stadtplans keine typische Anwendung für dieses Programm, aber es erstaunt mich immer wieder, wie einfach und vielfältig, wenn auch manchmal zeitaufwendig, die Arbeit mit GEOPAINT sein kann.


Berliner Strasse
Am Rathaus
30
32
28
26
24
22
20
19
17
18
15
13
11
16
14
9
7
10
5
3
1
Muehlen strasse
Am Rathaus
Muehlen strasse
Am Rathaus
Hochdahler Strasse
Muehlen strasse
Berliner Stasse
Am Rathaus
24
Am Rathaus

11. Anhänge

Anhang A Die Token-Referenzliste

Ich möchte Ihnen hier alle Basic-Befehle und deren Tokens auflisten. Da immer das 7. Bit gesetzt ist, ergibt sich immer ein Wert der größer 127 ist. Hier nun die Liste:

END	$80	128	$A831
FOR	$81	129	$A742
NEXT	$82	130	$AD1D
DATA	$83	131	$A8F8
INPUT#	$84	132	$ABA5
INPUT	$85	133	$ABBF
DIM	$86	134	$B081
READ	$87	135	$AC06
LET	$88	136	$A9A5
GOTO	$89	137	$A8A0
RUN	$8A	138	$A871
IF	$8B	139	$A928
RESTORE	$8C	140	$A81D
GOSUB	$8D	141	$A883
RETURN	$8E	142	$A8D2
REM	$8F	143	$A93B
STOP	$90	144	$A82F
ON	$91	145	$A94B
WAIT	$92	146	$B82D
LOAD	$93	147	$E168
SAVE	$94	148	$E156
VERIFY	$95	149	$E165
DEF	$96	150	$B3B3
POKE	$97	151	$B824
PRINT#	$98	152	$AA80
PRINT	$99	153	$AAA0
CONT	$9A	154	$A857
LIST	$9B	155	$A69C
CLR	$9C	156	$A65E
CMD	$9D	157	$AA86
SYS	$9E	158	$E12A
OPEN	$9F	159	$E1BE
CLOSE	$A0	160	$E1C7
GET/GET#	$A1	161	$AB7B
NEW	$A2	162	$A642
TAB	$A3	163	
TO	$A4	164	
FN	$A5	165	$B3F4
SPC	$A6	166	

THEN	$A7	167	
NOT	$A8	168	$AED4
STEP	$A9	169	
+	$AA	170	$B86A
-	$AB	171	$B853
*	$AC	172	$BA2B
/	$AD	173	$BB12
^	$AE	174	$BF7B
AND	$AF	175	$AFE9
OR	$B0	176	$AFE6
>	$B1	177	
=	$B2	178	
<	$B3	179	
SGN	$B4	180	$BC39
INT	$B5	181	$BCCC
ABS	$B6	182	$BC58
USR	$B7	183	$0310
FRE	$B8	184	$B37D
POS	$B9	185	$B39E
SQR	$BA	186	$BF71
RND	$BB	187	$E097
LOG	$BC	188	$B9EA
EXP	$BD	189	$BFED
COS	$BE	190	$E264
SIN	$BF	191	$E26B
TAN	$C0	192	$E2B4
ATN	$C1	193	$E30E
PEEK	$C2	194	$B80D
LEN	$C3	195	$B77C
STR$	$C4	196	$B465
VAL	$C5	197	$B7AD
ASC	$C6	198	$B78B
CHR$	$C7	199	$B6EC
LEFT$	$C8	200	$B700
RIGHT$	$C9	201	$B72C
MID$	$CA	202	$B737
GO	$CB	203	

Wie Sie der Tabelle entnehmen können, sind noch alle Werte ab 204 frei. Diese können Sie für eigene Befehle verwenden. Wie dies gemacht wird, erklärt das im DATA BECKER Verlag erschienene Buch "64 Intern" sehr gut.

Anhang B ASCII-Tabelle

Chr$ & Poke Kodes

Dez	Hex	POKE- Zeichen Gross	POKE- Zeichen Klein	CHR$- Zeichen Gross	CHR$- Zeichen Klein
000	$00	@	@	Keine Funktion	
001	$01	A	a	Keine Funktion	
002	$02	B	b	Keine Funktion	
003	$03	C	c	Keine Funktion	
004	$04	D	d	Keine Funktion	
005	$05	E	e	Weiss	
006	$06	F	f	Keine Funktion	
007	$07	G	g	Keine Funktion	
008	$08	H	h	Shift Commodore verboten	
009	$09	I	i	Shift Commodore erlauben	
010	$0a	J	j	Keine Funktion	
011	$0b	K	k	Keine Funktion	
012	$0c	L	l	Keine Funktion	
013	$0d	M	m	Return	
014	$0e	N	n	Kleinbuchstaben	
015	$0f	O	o	Keine Funktion	
016	$10	P	p	Keine Funktion	
017	$11	Q	q	Cursor abwaerts	
018	$12	R	r	Revers anschalten	
019	$13	S	s	Cursor Home	
020	$14	T	t	Delete	
021	$15	U	u	Keine Funktion	
022	$16	V	v	Keine Funktion	
023	$17	W	w	Keine Funktion	
024	$18	X	x	Keine Funktion	
025	$19	Y	y	Keine Funktion	
026	$1a	Z	z	Keine Funktion	
027	$1b	[	[	Escape	
028	$1c	£	£	Rot	
029	$1d	]	]	Cursor rechts	
030	$1e	↑	↑	Gruen	
031	$1f	←	←	Dunkelblau	
032	$20			Space	
033	$21	!	!	!	!
034	$22			"	"
035	$23	#	#	#	#
036	$24	$	$	$	$
037	$25	%	%	%	%
038	$26	&	&	&	&
039	$27	'	'	'	'
040	$28	(	(	(	(
041	$29	)	)	)	)
042	$2a	*	*	*	*
043	$2b	+	+	+	+
044	$2c	,	,	,	,
045	$2d	—	—	—	—
046	$2e	.	.	.	.
047	$2f	/	/	/	/
048	$30	0	0	0	0
049	$31	1	1	1	1
050	$32	2	2	2	2
051	$33	3	3	3	3
052	$34	4	4	4	4
053	$35	5	5	5	5
054	$36	6	6	6	6
055	$37	7	7	7	7
056	$38	8	8	8	8
057	$39	9	9	9	9
058	$3a	:	:	:	:
059	$3b	;	;	;	;

060	$3c	<	<	<	<
061	$3d	=	=	=	=
062	$3e	>	>	>	>
063	$3f	?	?	?	?
064	$40			@	@
065	$41		A	A	a
066	$42		B	B	b
067	$43		C	C	c
068	$44		D	D	d
069	$45		E	E	e
070	$46		F	F	f
071	$47		G	G	g
072	$48		H	H	h
073	$49		I	I	i
074	$4a		J	J	j
075	$4b		K	K	k
076	$4c		L	L	l
077	$4d		M	M	m
078	$4e		N	N	n
079	$4f		O	O	o
080	$50		P	P	p
081	$51		Q	Q	q
082	$52		R	R	r
083	$53		S	S	s
084	$54		T	T	t
085	$55		U	U	u
086	$56		V	V	v
087	$57		W	W	w
088	$58		X	X	x
089	$59		Y	Y	y
090	$5a		Z	Z	z
091	$5b			[	[
092	$5c			£	£
093	$5d			]	]
094	$5e			↑	↑
095	$5f			←	←
096	$60				
097	$61				A
098	$62				B
099	$63				C
100	$64				D
101	$65				E
102	$66				F
103	$67				G
104	$68				H
105	$69				I
106	$6a				J
107	$6b				K
108	$6c				L
109	$6d				M
110	$6e				N
111	$6f				O
112	$70				P
113	$71				Q
114	$72				R
115	$73				S
116	$74				T
117	$75				U
118	$76				V
119	$77				W
120	$78				X
121	$79				Y
122	$7a				Z
123	$7b				
124	$7c				
125	$7d				
126	$7e				
127	$7f				
128	$80				Keine Funktion
129	$81				Keine Funktion
130	$82				Keine Funktion
131	$83				Keine Funktion

132	$84			Keine Funktion	
133	$85			Weiss	
134	$86			Keine Funktion	
135	$87			Keine Funktion	
136	$88			Shift Commodore verboten	
137	$89			Shift Commodore erlauben	
138	$8a			Keine Funktion	
139	$8b			Keine Funktion	
14Ø	$8c			Keine Funktion	
141	$8d			Return	
142	$8e			Kleinbuchstaben	
143	$8f			Keine Funktion	
144	$9Ø			Keine Funktion	
145	$91			Cursor abwaerts	
146	$92			Revers anschalten	
147	$93			Cursor Home	
148	$94			Delete	
149	$95			Keine Funktion	
15Ø	$96			Keine Funktion	
151	$97			Keine Funktion	
152	$98			Keine Funktion	
153	$99			Keine Funktion	
154	$9a			Keine Funktion	
155	$9b			Escape	
156	$9c			Rot	
157	$9d			Cursor rechts	
158	$9e			Gruen	
159	$9f			Dunkelblau	
16Ø	$aØ			Space	
161	$a1			!	!
162	$a2			"	"
163	$a3			#	#
164	$a4			$	$
165	$a5			%	%
166	$a6			&	&
167	$a7			'	'
168	$a8			(	(
169	$a9			)	)
17Ø	$aa			*	*
171	$ab			+	+
172	$ac			,	,
173	$ad			—	—
174	$ae			.	.
175	$af			/	/
176	$bØ			Ø	Ø
177	$b1			1	1
178	$b2			2	2
179	$b3			3	3
18Ø	$b4			4	4
181	$b5			5	5
182	$b6			6	6
183	$b7			7	7
184	$b8			8	8
185	$b9			9	9
186	$ba			:	:
187	$bb			;	;
188	$bc			<	<
189	$bd			=	=
19Ø	$be			>	>
191	$bf			?	?
192	$cØ			@	@
193	$c1			A	a
194	$c2			B	b
195	$c3			C	c
196	$c4			D	d
197	$c5			E	e
198	$c6			F	f
199	$c7			G	g
2ØØ	$c8			H	h
2Ø1	$c9			I	i
2Ø2	$ca			J	j
2Ø3	$cb			K	k

2Ø4	$cc			L		l
2Ø5	$cd			M		m
2Ø6	$ce			N		n
2Ø7	$cf			O		o
2Ø8	$dØ			P		p
2Ø9	$d1			Q		q
21Ø	$d2			R		r
211	$d3			S		s
212	$d4			T		t
213	$d5			U		u
214	$d6			V		v
215	$d7			W		w
216	$d8			X		x
217	$d9			Y		y
218	$da			Z		z
219	$db			[		[
22Ø	$dc			£		£
221	$dd			]		]
222	$de			↑		↑
223	$df			←		←
224	$eØ			—		—
225	$e1					A
226	$e2					B
227	$e3					C
228	$e4					D
229	$e5					E
23Ø	$e6					F
231	$e7					G
232	$e8					H
233	$e9					I
234	$ea					J
235	$eb					K
236	$ec					L
237	$ed					M
238	$ee					N
239	$ef					O
24Ø	$fØ					P
241	$f1					Q
242	$f2					R
243	$f3					S
244	$f4					T
245	$f5					U
246	$f6					V
247	$f7					W
248	$f8					X
249	$f9					Y
25Ø	$fa					Z
251	$fb			+		+
252	$fc					
253	$fd					
254	$fe					
255	$ff					

Anhang C Der Zeichensatz

```
D000        (000)
3C □□■■■■□□
66 □■■□□■■□
6E □■■□■■■□
6E □■■□■■■□
60 □■■□□□□□
62 □■■□□□■□
3C □□■■■■□□
00 □□□□□□□□
```

```
D008        (001)
18 □□□■■□□□
3C □□■■■■□□
66 □■■□□■■□
7E □■■■■■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
00 □□□□□□□□
```

```
D010        (002)
7C □■■■■■□□
66 □■■□□■■□
66 □■■□□■■□
7C □■■■■■□□
66 □■■□□■■□
66 □■■□□■■□
7C □■■■■■□□
00 □□□□□□□□
```

```
D018        (003)
3C □□■■■■□□
66 □■■□□■■□
60 □■■□□□□□
60 □■■□□□□□
60 □■■□□□□□
66 □■■□□■■□
3C □□■■■■□□
00 □□□□□□□□
```

```
D020        (004)
78 □■■■■□□□
6C □■■□■■□□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
6C □■■□■■□□
78 □■■■■□□□
00 □□□□□□□□
```

```
D028        (005)
7E □■■■■■■□
60 □■■□□□□□
60 □■■□□□□□
78 □■■■■□□□
60 □■■□□□□□
60 □■■□□□□□
7E □■■■■■■□
00 □□□□□□□□
```

```
D030        (006)
7E □■■■■■■□
60 □■■□□□□□
60 □■■□□□□□
78 □■■■■□□□
60 □■■□□□□□
60 □■■□□□□□
60 □■■□□□□□
00 □□□□□□□□
```

```
D038        (007)
3C □□■■■■□□
66 □■■□□■■□
60 □■■□□□□□
6E □■■□■■■□
66 □■■□□■■□
66 □■■□□■■□
3C □□■■■■□□
00 □□□□□□□□
```

```
D040        (008)
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
7E □■■■■■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
00 □□□□□□□□
```

```
D048        (009)
3C □□■■■■□□
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
3C □□■■■■□□
00 □□□□□□□□
```

```
D050        (010)
1E □□□■■■■□
0C □□□□■■□□
0C □□□□■■□□
0C □□□□■■□□
0C □□□□■■□□
6C □■■□■■□□
38 □□■■■□□□
00 □□□□□□□□
```

```
D058        (011)
66 □■■□□■■□
6C □■■□■■□□
78 □■■■■□□□
70 □■■■□□□□
78 □■■■■□□□
6C □■■□■■□□
66 □■■□□■■□
00 □□□□□□□□
```

```
D060        (012)
60 □■■□□□□□
60 □■■□□□□□
60 □■■□□□□□
60 □■■□□□□□
60 □■■□□□□□
60 □■■□□□□□
7E □■■■■■■□
00 □□□□□□□□
```

```
D068        (013)
63 □■■□□□■■
77 □■■■□■■■
7F □■■■■■■■
6B □■■□■□■■
63 □■■□□□■■
63 □■■□□□■■
63 □■■□□□■■
00 □□□□□□□□
```

```
D070        (014)
66 □■■□□■■□
76 □■■■□■■□
7E □■■■■■■□
7E □■■■■■■□
6E □■■□■■■□
66 □■■□□■■□
66 □■■□□■■□
00 □□□□□□□□
```

```
D078        (015)
3C □□■■■■□□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
3C □□■■■■□□
00 □□□□□□□□
```

```
D080        (016)
7C □■■■■■□□
66 □■■□□■■□
66 □■■□□■■□
7C □■■■■■□□
60 □■■□□□□□
60 □■■□□□□□
60 □■■□□□□□
00 □□□□□□□□
```

```
D088        (017)
3C □□■■■■□□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
3C □□■■■■□□
0E □□□□■■■□
00 □□□□□□□□
```

```
D090        (018)
7C □■■■■■□□
66 □■■□□■■□
66 □■■□□■■□
7C □■■■■■□□
78 □■■■■□□□
6C □■■□■■□□
66 □■■□□■■□
00 □□□□□□□□
```

```
D098        (019)
3C □□■■■■□□
66 □■■□□■■□
60 □■■□□□□□
3C □□■■■■□□
06 □□□□□■■□
66 □■■□□■■□
3C □□■■■■□□
00 □□□□□□□□
```

```
D0A0        (020)
7E □■■■■■■□
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
00 □□□□□□□□
```

```
D0A8        (021)
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
3C □□■■■■□□
00 □□□□□□□□
```

```
D0B0        (022)
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
66 □■■□□■■□
3C □□■■■■□□
18 □□□■■□□□
00 □□□□□□□□
```

```
D0B8        (023)
63 □■■□□□■■
63 □■■□□□■■
63 □■■□□□■■
6B □■■□■□■■
7F □■■■■■■■
77 □■■■□■■■
63 □■■□□□■■
00 □□□□□□□□
```

D0C0 (024)	D0C8 (025)	D0D0 (026)	D0D8 (027)
66	66	7E	3C
66	66	06	30
3C	66	0C	30
18	3C	18	30
3C	18	30	30
66	18	60	30
66	18	7E	3C
00	00	00	00

D0E0 (028)	D0E8 (029)	D0F0 (030)	D0F8 (031)
0C	3C	00	00
12	0C	18	10
30	0C	3C	30
7C	0C	7E	7F
30	0C	18	7F
62	0C	18	30
FC	3C	18	10
00	00	18	00

D100 (032)	D108 (033)	D110 (034)	D118 (035)
00	18	66	66
00	18	66	66
00	18	66	FF
00	18	00	66
00	00	00	FF
00	00	00	66
00	18	00	66
00	00	00	00

D120 (036)	D128 (037)	D130 (038)	D138 (039)
18	62	3C	06
3E	66	66	0C
60	0C	3C	18
3C	18	38	00
06	30	67	00
7C	66	66	00
18	46	3F	00
00	00	00	00

D140 (040)	D148 (041)	D150 (042)	D158 (043)
0C	30	00	00
18	18	66	18
30	0C	3C	18
30	0C	FF	7E
30	0C	3C	18
18	18	66	18
0C	30	00	00
00	00	00	00

D160 (044)	D168 (045)	D170 (046)	D178 (047)
00	00	00	00
00	00	00	03
00	00	00	06
00	7E	00	0C
00	00	00	18
18	00	18	30
18	00	18	60
30	00	00	00

D180 (048)	D188 (049)	D190 (050)	D198 (051)
3C	18	3C	3C
66	18	66	66
6E	38	06	06
76	18	0C	1C
66	18	30	06
66	18	60	66
3C	7E	7E	3C
00	00	00	00

D1A0 (052)	D1A8 (053)	D1B0 (054)	D1B8 (055)
06	7E	3C	7E
0E	60	66	66
1E	7C	60	0C
66	06	7C	18
7F	06	66	18
06	66	66	18
06	3C	3C	18
00	00	00	00

D1C0 (056)	D1C8 (057)	D1D0 (058)	D1D8 (059)
3C	3C	00	00
66	66	00	00
66	66	18	18
3C	3E	00	00
66	06	00	00
66	66	18	18
3C	3C	00	18
00	00	00	30

D1E0 (060)	D1E8 (061)	D1F0 (062)	D1F8 (063)
0E	00	70	3C
18	00	18	66
30	7E	0C	06
60	00	06	0C
30	7E	0C	18
18	00	18	00
0E	00	70	18
00	00	00	00

D200 (064)	D208 (065)	D210 (066)	D218 (067)
00	08	18	00
00	1C	18	00
00	3E	18	00
FF	7F	18	FF
FF	7F	18	FF
00	1C	18	00
00	3E	18	00
00	00	18	00

D220 (068)	D228 (069)	D230 (070)	D238 (071)
00	00	00	30
00	FF	00	30
FF	FF	00	30
FF	00	00	30
00	00	FF	30
00	00	FF	30
00	00	00	30
00	00	00	30

D240 (072)	D248 (073)	D250 (074)	D258 (075)
0C □□□□■■□□	00 □□□□□□□□	18 □□□■■□□□	18 □□□■■□□□
0C □□□□■■□□	00 □□□□□□□□	18 □□□■■□□□	18 □□□■■□□□
0C □□□□■■□□	00 □□□□□□□□	1C □□□■■■□□	38 □□■■■□□□
0C □□□□■■□□	E0 ■■■□□□□□	0F □□□□■■■■	F0 ■■■■□□□□
0C □□□□■■□□	F0 ■■■■□□□□	07 □□□□□■■■	E0 ■■■□□□□□
0C □□□□■■□□	38 □□■■■□□□	00 □□□□□□□□	00 □□□□□□□□
0C □□□□■■□□	18 □□□■■□□□	00 □□□□□□□□	00 □□□□□□□□
0C □□□□■■□□	18 □□□■■□□□	00 □□□□□□□□	00 □□□□□□□□

D260 (076)	D268 (077)	D270 (078)	D278 (079)
C0 ■■□□□□□□	C0 ■■□□□□□□	03 □□□□□□■■	FF ■■■■■■■■
C0 ■■□□□□□□	E0 ■■■□□□□□	07 □□□□□■■■	FF ■■■■■■■■
C0 ■■□□□□□□	70 □■■■□□□□	0E □□□□■■■□	C0 ■■□□□□□□
C0 ■■□□□□□□	38 □□■■■□□□	1C □□□■■■□□	C0 ■■□□□□□□
C0 ■■□□□□□□	1C □□□■■■□□	38 □□■■■□□□	C0 ■■□□□□□□
C0 ■■□□□□□□	0E □□□□■■■□	70 □■■■□□□□	C0 ■■□□□□□□
FF ■■■■■■■■	07 □□□□□■■■	E0 ■■■□□□□□	C0 ■■□□□□□□
FF ■■■■■■■■	03 □□□□□□■■	C0 ■■□□□□□□	C0 ■■□□□□□□

D280 (080)	D288 (081)	D290 (082)	D298 (083)
FF ■■■■■■■■	00 □□□□□□□□	00 □□□□□□□□	36 □□■■□■■□
FF ■■■■■■■■	3C □□■■■■□□	00 □□□□□□□□	7F □■■■■■■■
03 □□□□□□■■	7E □■■■■■■□	00 □□□□□□□□	7F □■■■■■■■
03 □□□□□□■■	7E □■■■■■■□	00 □□□□□□□□	7F □■■■■■■■
03 □□□□□□■■	7E □■■■■■■□	00 □□□□□□□□	3E □□■■■■■□
03 □□□□□□■■	7E □■■■■■■□	FF ■■■■■■■■	1C □□□■■■□□
03 □□□□□□■■	3C □□■■■■□□	FF ■■■■■■■■	08 □□□□■□□□
03 □□□□□□■■	00 □□□□□□□□	00 □□□□□□□□	00 □□□□□□□□

D2A0 (084)	D2A8 (085)	D2B0 (086)	D2B8 (087)
60 □■■□□□□□	00 □□□□□□□□	C3 ■■□□□□■■	00 □□□□□□□□
60 □■■□□□□□	00 □□□□□□□□	E7 ■■■□□■■■	3C □□■■■■□□
60 □■■□□□□□	00 □□□□□□□□	7E □■■■■■■□	7E □■■■■■■□
60 □■■□□□□□	07 □□□□□■■■	3C □□■■■■□□	66 □■■□□■■□
60 □■■□□□□□	0F □□□□■■■■	3C □□■■■■□□	66 □■■□□■■□
60 □■■□□□□□	1C □□□■■■□□	7E □■■■■■■□	7E □■■■■■■□
60 □■■□□□□□	18 □□□■■□□□	E7 ■■■□□■■■	3C □□■■■■□□
60 □■■□□□□□	18 □□□■■□□□	C3 ■■□□□□■■	00 □□□□□□□□

D2C0 (088)	D2C8 (089)	D2D0 (090)	D2D8 (091)
18 □□□■■□□□	06 □□□□□■■□	08 □□□□■□□□	18 □□□■■□□□
18 □□□■■□□□	06 □□□□□■■□	1C □□□■■■□□	18 □□□■■□□□
66 □■■□□■■□	06 □□□□□■■□	3E □□■■■■■□	18 □□□■■□□□
66 □■■□□■■□	06 □□□□□■■□	7F □■■■■■■■	FF ■■■■■■■■
18 □□□■■□□□	06 □□□□□■■□	3E □□■■■■■□	FF ■■■■■■■■
18 □□□■■□□□	06 □□□□□■■□	1C □□□■■■□□	18 □□□■■□□□
3C □□■■■■□□	06 □□□□□■■□	08 □□□□■□□□	18 □□□■■□□□
00 □□□□□□□□	06 □□□□□■■□	00 □□□□□□□□	18 □□□■■□□□

D2E0 (092)	D2E8 (093)	D2F0 (094)	D2F8 (095)
C0 ■■□□□□□□	18 □□□■■□□□	00 □□□□□□□□	FF ■■■■■■■■
C0 ■■□□□□□□	18 □□□■■□□□	00 □□□□□□□□	7F □■■■■■■■
30 □□■■□□□□	18 □□□■■□□□	03 □□□□□□■■	3F □□■■■■■■
30 □□■■□□□□	18 □□□■■□□□	3E □□■■■■■□	1F □□□■■■■■
C0 ■■□□□□□□	18 □□□■■□□□	76 □■■■□■■□	0F □□□□■■■■
C0 ■■□□□□□□	18 □□□■■□□□	36 □□■■□■■□	07 □□□□□■■■
30 □□■■□□□□	18 □□□■■□□□	36 □□■■□■■□	03 □□□□□□■■
30 □□■■□□□□	18 □□□■■□□□	00 □□□□□□□□	01 □□□□□□□■

```
D300          (096)

00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
```

```
D308          (097)

F0 ■■■■□□□□
F0 ■■■■□□□□
F0 ■■■■□□□□
F0 ■■■■□□□□
F0 ■■■■□□□□
F0 ■■■■□□□□
F0 ■■■■□□□□
F0 ■■■■□□□□
```

```
D310          (098)

00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
FF ■■■■■■■■
FF ■■■■■■■■
FF ■■■■■■■■
FF ■■■■■■■■
```

```
D318          (099)

FF ■■■■■■■■
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
```

```
D320          (100)

00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
FF ■■■■■■■■
```

```
D328          (101)

C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
```

```
D330          (102)

CC ■■□□■■□□
CC ■■□□■■□□
33 □□■■□□■■
33 □□■■□□■■
CC ■■□□■■□□
CC ■■□□■■□□
33 □□■■□□■■
33 □□■■□□■■
```

```
D338          (103)

03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
```

```
D340          (104)

00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
CC ■■□□■■□□
CC ■■□□■■□□
33 □□■■□□■■
33 □□■■□□■■
```

```
D348          (105)

FF ■■■■■■■■
FE ■■■■■■■□
FC ■■■■■■□□
F8 ■■■■■□□□
F0 ■■■■□□□□
E0 ■■■□□□□□
C0 ■■□□□□□□
80 ■□□□□□□□
```

```
D350          (106)

03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
03 □□□□□□■■
```

```
D358          (107)

18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
1F □□□■■■■■
1F □□□■■■■■
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
```

```
D360          (108)

00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
0F □□□□■■■■
0F □□□□■■■■
0F □□□□■■■■
0F □□□□■■■■
```

```
D368          (109)

18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
1F □□□■■■■■
1F □□□■■■■■
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
```

```
D370          (110)

00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
F8 ■■■■■□□□
F8 ■■■■■□□□
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
```

```
D378          (111)

00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
FF ■■■■■■■■
FF ■■■■■■■■
```

```
D380          (112)

00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
1F □□□■■■■■
1F □□□■■■■■
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
```

```
D388          (113)

18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
FF ■■■■■■■■
FF ■■■■■■■■
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
```

```
D390          (114)

00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
FF ■■■■■■■■
FF ■■■■■■■■
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
```

```
D398          (115)

18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
F8 ■■■■■□□□
F8 ■■■■■□□□
18 □□□■■□□□
18 □□□■■□□□
18 □□□■■□□□
```

```
D3A0          (116)

C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
C0 ■■□□□□□□
```

```
D3A8          (117)

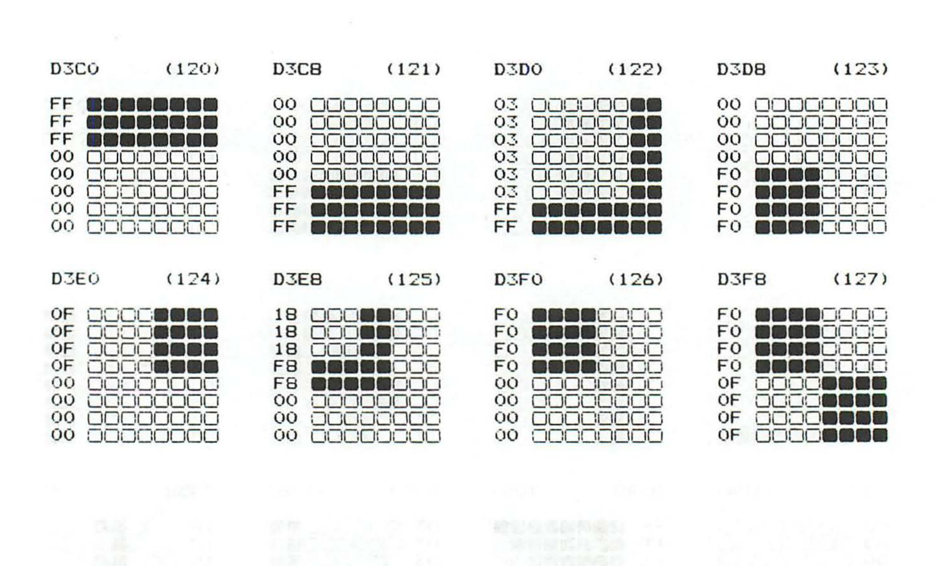
E0 ■■■□□□□□
E0 ■■■□□□□□
E0 ■■■□□□□□
E0 ■■■□□□□□
E0 ■■■□□□□□
E0 ■■■□□□□□
E0 ■■■□□□□□
E0 ■■■□□□□□
```

```
D3B0          (118)

07 □□□□□■■■
07 □□□□□■■■
07 □□□□□■■■
07 □□□□□■■■
07 □□□□□■■■
07 □□□□□■■■
07 □□□□□■■■
07 □□□□□■■■
```

```
D3B8          (119)

FF ■■■■■■■■
FF ■■■■■■■■
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
00 □□□□□□□□
```


D3C0 (120)
FF FF FF 00 00 00 00 00
D3C8 (121)
00 00 00 00 00 FF FF FF
D3D0 (122)
03 03 03 03 03 03 FF FF
D3D8 (123)
00 00 00 00 F0 F0 F0 F0
D3E0 (124)
0F 0F 0F 0F 00 00 00 00
D3E8 (125)
18 18 18 F8 F8 00 00 00
D3F0 (126)
F0 F0 F0 F0 00 00 00 00
D3F8 (127)
F0 F0 F0 F0 0F 0F 0F 0F

Anhang D Umrechnungstabelle

Da es immer recht umständlich ist, Zahlen von dem einen Zahlensystem ins andere umzurechnen, habe ich Ihnen hier ein Umrechnungstabelle zusammengestellt, die die drei Zahlensysteme beinhaltet:

Dezimal	Hexadezimal	Binär
000	$00	00000000
001	$01	00000001
002	$02	00000010
003	$03	00000011
004	$04	00000100
005	$05	00000101
006	$06	00000110
007	$07	00000111
008	$08	00001000
009	$09	00001001
010	$0A	00001010
011	$0B	00001011
012	$0C	00001100
013	$0D	00001101
014	$0E	00001110
015	$0F	00001111
016	$10	00010000
017	$11	00010001
018	$12	00010010
019	$13	00010011
020	$14	00010100
021	$15	00010101
022	$16	00010110
023	$17	00010111
024	$18	00011000
025	$19	00011001
026	$1A	00011010
027	$1B	00011011
028	$1C	00011100
029	$1D	00011101
030	$1E	00011110
031	$1F	00011111
032	$20	00100000
033	$21	00100001
034	$22	00100010
035	$23	00100011

036	$24	00100100
037	$25	00100101
038	$26	00100110
039	$27	00100111
040	$28	00101000
041	$29	00101001
042	$2A	00101010
043	$2B	00101011
044	$2C	00101100
045	$2D	00101101
046	$2E	00101110
047	$2F	00101111
048	$30	00110000
049	$31	00110001
050	$32	00110010
051	$33	00110011
052	$34	00110100
053	$35	00110101
054	$36	00110110
055	$37	00110111
056	$38	00111000
057	$39	00111001
058	$3A	00111010
059	$3B	00111011
060	$3C	00111100
061	$3D	00111101
062	$3E	00111110
063	$3F	00111111
064	$40	01000000
065	$41	01000001
066	$42	01000010
067	$43	01000011
068	$44	01000100
069	$45	01000101
070	$46	01000110
071	$47	01000111
072	$48	01001000
073	$49	01001001
074	$4A	01001010
075	$4B	01001011
076	$4C	01001100
077	$4D	01001101
078	$4E	01001110
079	$4F	01001111
080	$50	01010000
081	$51	01010001

082	$52	01010010
083	$53	01010011
084	$54	01010100
085	$55	01010101
086	$56	01010110
087	$57	01010111
088	$58	01011000
089	$59	01011001
090	$5A	01011010
091	$5B	01011011
092	$5C	01011100
093	$5D	01011101
094	$5E	01011110
095	$5F	01011111
096	$60	01100000
097	$61	01100001
098	$62	01100010
099	$63	01100011
100	$64	01100100
101	$65	01100101
102	$66	01100110
103	$67	01100111
104	$68	01101000
105	$69	01101001
106	$6A	01101010
107	$6B	01101011
108	$6C	01101100
109	$6D	01101101
110	$6E	01101110
111	$6F	01101111
112	$70	01110000
113	$71	01110001
114	$72	01110010
115	$73	01110011
116	$74	01110100
117	$75	01110101
118	$76	01110110
119	$77	01110111
120	$78	01111000
121	$79	01111001
122	$7A	01111010
123	$7B	01111011
124	$7C	01111100
125	$7D	01111101
126	$7E	01111110
127	$7F	01111111

128	$80	10000000
129	$81	10000001
130	$82	10000010
131	$83	10000011
132	$84	10000100
133	$85	10000101
134	$86	10000110
135	$87	10000111
136	$88	10001000
137	$89	10001001
138	$8A	10001010
139	$8B	10001011
140	$8C	10001100
141	$8D	10001101
142	$8E	10001110
143	$8F	10001111
144	$90	10010000
145	$91	10010001
146	$92	10010010
147	$93	10010011
148	$94	10010100
149	$95	10010101
150	$96	10010110
151	$97	10010111
152	$98	10011000
153	$99	10011001
154	$9A	10011010
155	$9B	10011011
156	$9C	10011100
157	$9D	10011101
158	$9E	10011110
159	$9F	10011111
160	$A0	10100000
161	$A1	10100001
162	$A2	10100010
163	$A3	10100011
164	$A4	10100100
165	$A5	10100101
166	$A6	10100110
167	$A7	10100111
168	$A8	10101000
169	$A9	10101001
170	$AA	10101010
171	$AB	10101011
172	$AC	10101100
173	$AD	10101101

174	$AE	10101110
175	$AF	10101111
176	$B0	10110000
177	$B1	10110001
178	$B2	10110010
179	$B3	10110011
180	$B4	10110100
181	$B5	10110101
182	$B6	10110110
183	$B7	10110111
184	$B8	10111000
185	$B9	10111001
186	$BA	10111010
187	$BB	10111011
188	$BC	10111100
189	$BD	10111101
190	$BE	10111110
191	$BF	10111111
192	$C0	11000000
193	$C1	11000001
194	$C2	11000010
195	$C3	11000011
196	$C4	11000100
197	$C5	11000101
198	$C6	11000110
199	$C7	11000111
200	$C8	11001000
201	$C9	11001001
202	$CA	11001010
203	$CB	11001011
204	$CC	11001100
205	$CD	11001101
206	$CE	11001110
207	$CF	11001111
208	$D0	11010000
209	$D1	11010001
210	$D2	11010010
211	$D3	11010011
212	$D4	11010100
213	$D5	11010101
214	$D6	11010110
215	$D7	11010111
216	$D8	11011000
217	$D9	11011001
218	$DA	11011010
219	$DB	11011011

220	$DC	11011100
221	$DD	11011101
222	$DE	11011110
223	$DF	11011111
224	$E0	11100000
225	$E1	11100001
226	$E2	11100010
227	$E3	11100011
228	$E4	11100100
229	$E5	11100101
230	$E6	11100110
231	$E7	11100111
232	$E8	11101000
233	$E9	11101001
234	$EA	11101010
235	$EB	11101011
236	$EC	11101100
237	$ED	11101101
238	$EE	11101110
239	$EF	11101111
240	$F0	11110000
241	$F1	11110001
242	$F2	11110010
243	$F3	11110011
244	$F4	11110100
245	$F5	11110101
246	$F6	11110110
247	$F7	11110111
248	$F8	11111000
249	$F9	11111001
250	$FA	11111010
251	$FB	11111011
252	$FC	11111100
253	$FD	11111101
254	$FE	11111110
255	$FF	11111111

Anhang E Glossar

Das Glossar ist zum Nachschlagen da, falls Ihnen einmal die Bedeutung eines Wortes nicht ganz klar ist. Hier werden die wichtigsten Fachwörter erklärt.

Adresse

Unter einer Adresse versteht man in der Computersprache genau das gleiche wie unter einer 'normalen' Adresse. Da der Computer auf verschiedene Speicherstellen zugreifen muß, muß er sie irgendwie unterscheiden können. Dazu hat jede Speicherstellen eine individuelle Nummer. Diese Nummer nennt man Adresse.

ASCII-Code

ASCII steht für 'American Standard Code for International Interchange'. Da der Rechner intern nur mit Zahlen, nicht aber mit Zeichen arbeiten kann, hat jeder Buchstabe und jedes andere Zeichen einen Code, den sogenannten ASCII-Code, der sich standardisiert hat.

Assembler

Der Rechner versteht nur Zahlen, nicht jedoch Hochsprachen wie BASIC etc. Assembler ist die maschinennächste Sprache. Es gibt nur sehr simple Befehle, mit denen auch nur Werte in Speicherstellen verändert werden können. Deshalb ist Assembler sehr umständlich zu programmieren, dafür jedoch extrem schnell.

Autostart

Normalerweise muß ein Programm, nachdem es geladen wurde, mittels eines Befehles gestartet werden. Ist das Programm mit einem Autostart versehen, so wird es automatisch nach dem Laden gestartet.

Betriebssystem

Das Betriebssystem ist nichts anderes als ein Programm, das für die elementarsten Dinge im Computer zuständig ist. Es regelt z.B. die Ein- und Ausgabe, die Kommunikation mit den Peripheriegeräten und vieles mehr. Das Betriebssystem ist in ein Eprom gebrannt.

Binärsystem

Darstellung eines Zahlensystems, das nur aus 0 und 1 besteht. Das Binärsystem wird auch Dualsystem genannt. Weiters siehe unter Zahlensysteme.

Bit

Elementarste Einheit eines Rechners. Ein Bit kann nur angeschaltet (Strom fließt; Wert: 1) oder ausgeschaltet (Strom fließt nicht; Wert: 0) sein.

Bitoperation

Unter Bitoperation versteht man das gezielte Setzen oder Löschen eines Bits innerhalb eines Bytes. Dazu gibt es die Befehle AND, OR und NOT. Der Befehl AND setzt genau dann ein Bit, wenn die Bits der Vergleichsoperanten beide 1 waren. Hierzu ein Beispiel:

```
Vergleichsoperand 1   00101001    41    29
Vergleichsoperand 2   10011011   155    9B
AND                   ________   ___    __
Ergebnis              00001001     9    09
```

Durch den AND-Befehl können Sie gezielt Bits löschen, indem Sie den zu verändernden Wert mit einer Zahl verknüpfen, in der nur die Bits nicht gesetzt sind, die gelöscht werden sollen. Beispiel: 10010101. Das dritte und fünfte Bit soll gelöscht werden:

	10010101	149	95
	11101011	235	EB
AND	———	—	—
	10000001	129	81

Mittels des OR-Befehls werden die Bits im Ergebnis gesetzt, die entweder im ersten oder im zweiten oder in beiden Vergleichsoperanden gesetzt waren. Nur wenn beide Bits null waren, wird dies im Ergebnis nicht gesetzt.

Vergleichsoperand 1	00101001	41	29
Vergleichsoperand 2	10011011	155	9B
OR	———	—	—
Ergebnis	10111011	187	BB

Mittels des OR-Befehls können Bits gesetzt werden. Sie müssen im zweiten Operanden die Bits setzen, die hinterher im Ergebnis gesetzt werden sollen. Beispiel: 10010100. Das 0., 1. und 6. Bit soll gesetzt werden:

	10010100	148	94
	01000011	67	43
OR	———	—	—
	11010111	215	D7

Des weiteren gibt es noch den NOT-Befehl. Dieser dreht einfach die Bits um; war also ein Bit gesetzt, so wird es gelöscht und umgekehrt. Auch hier ein Beispiel:

Vergleichsoperand 1	00111010	58	3A
NOT	———	—	—
Ergebnis	11000101	197	C5

Näheres in Kapitel 4.

Byte

Ein Byte besteht aus acht Bits. Es herrscht folgende Anordnung:

Bit	7	6	5	4	3	2	1	0
Wert	128	64	32	16	8	4	2	1

Ein Byte kann also Werte zwischen 0 und 255 annehmen.

Compiler

Ein Compiler ist ein Programm, das eine Hochsprache wie z.B. Basic oder C in Maschinensprache übersetzt. Dadurch werden die Programme im Ablauf schneller (to compile=zusammenstellen/übertragen).

Directory

Unter einem Directory versteht man das Inhaltsverzeichnis einer Diskette.

Dualsystem

Siehe Binärsystem.

Expansionsport

Der Expansionsport ist eine 'Buchse' an der Rückseite Ihres Rechners, an der Module und andere Erweiterungen angeschlossen werden können.

File

Ein File ist ein Programm oder eine Datei, die sich auf der Diskette oder Kassette befindet (File=Akte).

GEOS

GEOS ist eine Abkürzung für Graphic Environment Operating System. Dies ist ein neues Betriebssystem, das mit grafischen Symbolen statt mit Befehlssequenzen arbeitet und mittels Maus oder Joystick gesteuert wird (ähnlich dem GEM).

Geräteadresse

Da der Rechner verschiedene Geräte verwalten muß, hat jedes Gerät eine eigene Adresse, durch die sie angesprochen werden können. So besitzt die Floppy z.B. die Geräteadresse 8, der Drucker die Geräteadresse 4 usw. Ein Tabelle befindet sich in Kapitel 4.

Grafikmodus

Der Grafikmodus ist ein Modus, in dem jeder Punkt einzeln angesprochen werden kann. Dadurch können Grafiken in hoher Auflösung erzeugt werden. Es ist unter normalen Umständen nicht möglich, Text im Grafikmodus auf den Bildschirm zu bringen.

Hexadezimalsystem

Zahlensystem, das zur Basic 16 arbeitet. Die Zahlen 10-15 werden durch die Buchstaben A-F dargestellt. Siehe auch unter Zahlensysteme.

Interpreter

Ein Interpreter ist ein Programm, das eine Hochsprache in Maschinensprache übersetzt. Dies geschieht jedoch im Gegensatz zum Compiler während des Programmablaufes, wodurch das Programm entsprechend langsamer wird, da jeder Befehl wieder neu übersetzt wird. Im C64 ist ein Basicinterpreter in einem Eprom eingebrannt.

Interrupt

Alle 50tel Sekunde wird ein sogenannter Interrupt ausgeführt, was nichts anderes als eine Unterbrechung ist. Es wird ein Routine angesprungen, die verschiedene Zustände überprüft. Diese Routine heißt Interrupt-Routine. Sie ist z.B. für das Blinken des Cursors zuständig. Mit etwas Erfahrung läßt sich die Interrupt-Routine selbst neu programmieren.

Joystick

Steuerknüppel, der über einen der beiden Joystickports auf der linken Seite an den Rechner angeschlossen wird. Durch Bewegen dieses Knüppels können z.B. bei einem Spiel Männchen bewegt werden. Auch der Mauszeiger bei GEOS läßt sich durch den Joystick steuern.

Maus

Eingabegerät, das genauso wie der Joystick mit dem Rechner verbunden ist. Durch Bewegung der Maus auf einer Oberfläche entstehen Impulse, die zum Rechner weitergeleitet und dort ausgewertet werden. GEOS ist durch die Maus steuerbar.

Nibble

Ein Nibble sind 4 Bits. Es gibt das sogenannte Low-Nibble, das aus den Bits 0-3 besteht. Das sogenannte High-Nibble besteht aus den Bits 4-7.

Parallel

Art der Datenübertragung. Bei der Parallelübertragung werden die einzelnen Bits nebeneinander, also parallel übertragen. Dadurch findet die Übertragung schneller statt. Viele Floppyspeeder arbeiten nach diesem Prinzip (SpeedDOS).

Peripherie

Sammelbegriff für alle am Computer anschließbaren Geräte.

Prozessor

Herz des Rechners. Der Prozessor erledigt alle anfallenden Arbeiten und steuert die Zusatzbausteine.

RAM

Random Access Memory. Das RAM ist ein Speicherbereich, der sowohl beschrieben als auch gelesen werden kann. Ein Teil des RAMs wird vom BASIC-Speicher belegt.

ROM

Read Only Memory. Das ROM ist ein Speicherbereich, aus dem nur gelesen werden kann. Das ROM ist also nicht veränderbar. Im ROM befindet sich z.B. das Betriebssystem.

Scrolling

Das Wort Scolling ist ein zusammengesetztes Wort. Es besteht aus den Worten Screen und Rolling, was zusammen nichts anderes als Bildschirmrollen heißt. Durch Scrollen wird wird also der gesamte Bildschirm in eine Richtung bewegt. Viele Spiele beinhalten Scrolling.

Sektor

Abschnitt auf einer Diskette, der 256 Bytes beinhaltet.

Sekundäradresse

Die Sekundäradresse ist eine Zahl, die bei der Kommunikation mit Peripheriegeräten angegeben werden kann. Was diese Nummer bewirkt, hängt von dem Gerät ab. Gibt man z.B. bei der

Floppy die Sekundäradresse 1 an, so wird das Programm an die Adresse geladen, an der es auch beim Abspeichern stand.

Seriell

Art der Datenübertragung, bei der die Bits nacheinander übertragen werden. Diese Art der Datenübertragung ist deshalb auch entsprechend langsam.

Sprite

Ein Sprite ist eine Grafik mit einer Abmessung von 21*24 Pixel. Das Besondere daran ist, daß diese Grafik beliebig über den Bildschirm bewegt werden kann, ohne den Hintergrund zu zerstören. Sprites sind unabhängig vom Modus, können also auch im Textmodus dargestellt werden.

Spur

Eine Spur ist ein Abschnitt auf der Diskette, der genau eine Umdrehung lang ist. Eine Diskette beinhaltet (normal formatiert) 35 Spuren. Eine Spur besteht aus mehreren Sektoren.

Steuerzeichen

Unter einem Steuerzeichen versteht man ein Zeichen, das bei dessen Ausgabe einen bestimmten Vorgang auslöst. So gibt es beispielsweise ein Steuerzeichen, um den Bildschirm zu löschen, und eines, um die Klein-/Groß-Umschaltung zu unterbinden.

Token

Unter einem Token versteht man eine Zahl, die für einen bestimmten Basic-Befehl steht. Jeder Befehl besitzt so ein Zeichen. Statt den Befehl abzuspeichern, wird nur das Token abgespeichert, wodurch Speicherplatz und Diskettenplatz gespart wird (siehe Tabelle im Anhang A).

Unterprogramm

Ein Unterprogramm ist ein Programm, das eine bestimmte Arbeit erledigt und von allen Stellen des Hauptprogrammes aus aufgerufen werden kann. Oft wird auch Routine statt Unterprogramm gesagt.

Variable

Platzhalter für eine Zahl oder eine Zeichenkette. Variablen für Zahlenketten müssen die Endung $ erhalten. So ist es möglich, zwischen einer Textvariablen und einer Zahlenvariablen zu unterscheiden.

Zahlensysteme

Es gibt drei wichtige Zahlensysteme in der 'Computerei'. Das erste wäre das sogenannte Dezimalsystem, das die Basis 10 besitzt und allen bekannt sein sollte. Das Dezimalsystem wird hauptsächlich in Hochsprachen (BASIC, Pascal) benutzt. Das zweite ist das Hexadezimalsystem, das die Basis 16 besitzt. Dabei werden die 'Zahlen' 10-15 durch die Buchstaben A-F ersetzt. Das Dualsystem ist für die Maschinensprache enorm wichtig, genauso wie das Dual- oder Binärsystem, das als Basis die Zahl 2 hat.

Zeiger

Ein Zeiger besteht aus zwei Bytes, dem sogenannten Low- und High-Byte. Ein Zeiger zeigt dem Betriebssystem an, zu welcher Speicherstelle es verzweigen soll. Die Umrechnung von Low- und High-Byte erfolgt durch folgende Formel:

```
Adresse = Low-Byte + High-Byte*256
```

Umgekehrt läßt sich die Adresse durch folgende Formel in Low- und High-Byte zerlegen:

```
High-Byte = Adresse/256
Low-Byte  = Adresse - High-Byte
```

Es wird immer zuerst das Low-Byte, dann das High-Byte gespeichert.

Zeropage

Der Speicherbereich von 0 bis 255. In diesem Speicherbereich befinden sich viele wichtige Zeiger, die leicht veränderbar sind. Deshalb kann durch Verändern von Adressen in der Zeropage auch viel erreicht werden.

12. Stichwortverzeichnis

Vom Einsteiger bis zum Profi - an vielen Beispielen wird gezeigt, wie man mit GEOS wirklich kreativ arbeiten kann. Und nicht nur das: Darüber hinaus gibt es eine Vielzahl von Tips und Tricks zur Arbeit mit GEOS, was komfortables Erstellen eigener Programme und interessante Erweiterungsmöglichkeiten angeht. Wichtige Informationen über die Funktionsweise von GEOS, zu Einsprungadressen und Parameterübergabe komplettieren dieses Buch.



Aus dem Inhalt:

- Umfassende Einführung in die Arbeit mit GEOS
- Eigene Programme mit allen Eigenschaften von GEOS (ICONs, Info-Fenster, Hilfstexte)
- Echtzeituhr ständig im Bild
- Einzelschrittsimulator
- Alarmton und zusätzliche Bildschirmeffekte
- Fenstertechnik mit einem einzigen Befehl ausnutzen
- Maussteuerung durch den IRQ
- Das neue GEOS-Fileformat
- Speicheraufteilung und die wichtigsten Speicherstellen

Kerkloh, Tornsdorf
Das große GEOS-Buch
Hardcover, 350 Seiten, DM 49,—
ISBN 3-89011-208-0

Bücher zum Commodore 64

Sie wollen die vielfältigen Möglichkeiten Ihres neuerworbenen C64 möglichst schnell kennenlernen. Der Schlüssel dazu heißt BASIC. Daß die Einarbeitung darin nicht schwierig sein muß, ja sogar Spaß machen kann, zeigt dieses Buch. Schritt für Schritt werden Sie in die Welt des C64 und seines BASIC eingeführt. Durch Zusammenfassungen werden die neuerworbenen Fähigkeiten zum sicheren Besitz. Das umfangreiche Lexikon am Schluß macht das Buch für Sie auch noch wertvoll, wenn Sie kein Einsteiger mehr sind.



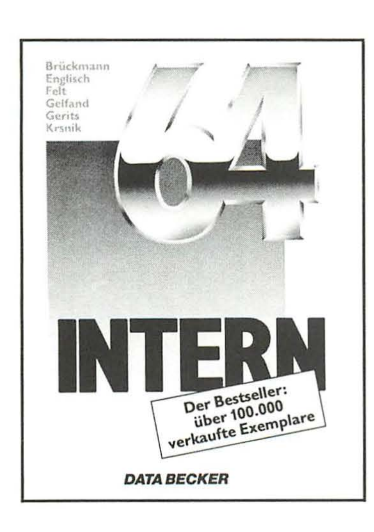
Aus dem Inhalt:

- BASIC an einem Abend: Der schnelle Weg zum ersten Programm
- BASIC an einem Wochenende: Variablen (Bedeutung und Arten, Anwendungsbeispiele aus der alltäglichen Praxis)
- Grundbefehle für die Erstellung von Programmabläufen – GOTO, IF THEN, FOR NEXT
- Der Weg zum Profi-Programm – Einsatz von Unterprogrammen
- Zahlreiche Anwendungsbeispiele für Programme
- Grafik zum Nulltarif: einfache Grafik (von Phantasiebildern bis hin zu Balkendiagrammen)
- Datenspeicherung innerhalb von Programmen und auf Diskette – sequentielle Dateien
- Hochauflösende Grafik
- SPRITES und dazu ein Spielprogramm (Landung eines Ballons)
- Pannenhilfe – Zusammenstellung von Hilfen bei Problemen
- Lexikon – umfangreiche, leichtverständliche Begriffserklärungen zum Nachschlagen

Tornsdorf
C64 BASIC für Einsteiger
246 Seiten, DM 29,–
ISBN 3-89011-246-3

Bücher zum Commodore 64

64 Intern ist ein Standardwerk zum Commodore 64, das vom ausführlich dokumentierten ROM-Listing über die detaillierte Hardwarebeschreibung bis zu nützlichen BASIC-Erweiterungen alles enthält, was man zum professionellen Einsatz des Commodore 64 wissen muß.



Aus dem Inhalt:

- Speicherbelegungsplätze
- Der Soundcontroller und seine Programmierung
- Die Handhabung des AD-Wandlers
- Der Videocontroller
- Programmierung von Farbe und Grafik
- Die Zeichengenerator-Schnittstelle
- Sprites
- Ein-/Ausgabesteuerung
- Timer und Echtzeituhr
- Joystickprogrammierung
- So arbeitet der BASIC-Interpreter
- Matematische Routinen - selbst entwickelt
- Der serielle IEC-Bus
- Programmierung der RS-232
- Die Belegung der Zero-Page
- Der Commodore-64-Schaltplan

Brückmann, Englisch, Felt, Gelfand, Gerits, Krsnik
64 Intern
Hardcover, 628 Seiten, DM 69,—
ISBN 3-89011-000-2
Inkl. Diskette!
Lieferbar ab 8/88.

[illegible] Bücher zum Commodore 64

64 Intern ist ein Standardwerk zum Commodore 64, das vom ausführlich dokumentierten ROM-Listing über die detaillierte Hardwarebeschreibung bis zu nützlichen BASIC-Erweiterungen alles enthält, was man zum professionellen Einsatz des Commodore 64 wissen muß.

Aus dem Inhalt:

- [illegible]
- Die Commodore [illegible]
- Programmierung
- Die Handhabung [illegible]
- Der [illegible]
- Programmierung von Farbe und Grafik
- Die Zeichendarstellung [illegible]
- Sprites
- Die Registerbelegung
- Timer und Echtzeituhr
- Joystickprogrammierung
- So arbeitet der BASIC-Interpreter
- [illegible]
- Der serielle IEC-Bus
- Programmierung der RS 232
- Die Belegung der Zeropage
- Der Commodore 64-Schaltplan



Brückmann, Englisch, Felt, Gelfand, Gerits, Krsnik
64 Intern
Hardcover, 628 Seiten, DM 69,–
ISBN 3-89011-000-2
inkl. Diskette!
Lieferbar ab 3/88.

Ein Grafikbetriebssystem für Ihren C64 – das bietet Ihnen dieses Buch. Und natürlich die Möglichkeit, wirklich alles über die C64-Grafik zu lernen. Der Schwierigkeitsgrad ist dabei frei wählbar: vom Einsteigen mit Hunderten von neuen Befehlen bis zum Programmieren von Grafikalgorithmen in Assembler. Vom CIRCLE-Befehl bis zu seinem kommentierten Source-Code.



Aus dem Inhalt:

- Multicolorgrafik
- CAD-Zeichner – das Zeichenpaket
- Grafikfenster durch Raster-Interrupt
- Text in hochauflösender Grafik
- Shapes auf dem C64
- 136 Farben auf dem C64
- Einladen von Fremdgrafiken
- Viele Utilities
- 3-D-Grafik
- Animation und Scrolling
- Text und Grafik auf dem Low-Res-Bildschirm
- Kommentiertes Source-Listing von Supergrafik

Plenge
Das Supergrafikbuch zum C64
Hardcover, mit Diskette im Buch, 726 Seiten, DM 49,–
ISBN 3-89011-213-7

Bücher zum Commodore 64

Vom Einsteiger bis zum Softwarehaus – dieses Buch zeigt, wie man seine Programme optimal schützt. Dabei werden Programm- und Kopierschutzverfahren vom einfachen List-Schutz bis zur Überprüfung eines kompletten Tracks gezeigt. Und: Die Autoren sind nicht bei bekannten Schutzmechanismen der neuesten Generation stehengeblieben, sondern haben neue entwickelt, für die es bisher keine Kopiermöglichkeiten gibt. Das Buch zeigt nicht nur die Verfahren, sondern liefert auch gleich fertige Lösungen für C64 und C128 zum Abtippen.

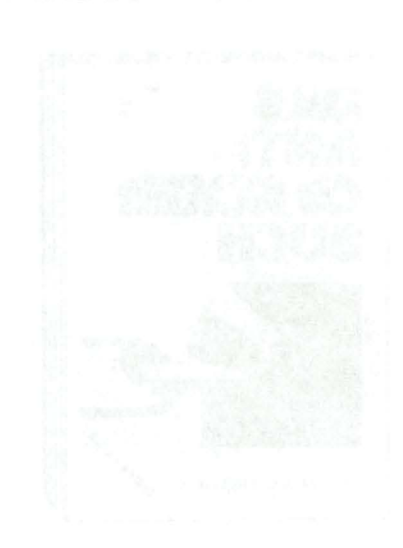


Aus dem Inhalt:

- BASIC-Programme schützen (List-Schutz, Änderungsschutz, Password, RESTORE/RESET-Schutz, SYS-Bluff, Compiler-Schutz)
- Alle Tricks des Autostart (Tastaturpuffer, Sprungvektoren, Stack, Interrupt, Adreßverschiebung beim Laden)
- Prüfsummen und Selbstzerstörung (XOR-Codierung, Timer-Codierung, Einzelschrittcodierung, ASCII-Codierung)
- Illegale Opcodes
- Cassettenkopierschutz (EOF, Cassettenpuffer, Autostart, Schnelladeverfahren, komplettes Kopierschutzsystem)
- Directory-Schutz (versteckter Filename, "blinde" Blöcke, der Nullen-Trick, geteiltes Directory, gelöschte Files)
- Grundlagen des Kopierschutzes
- Alle Tricks der Formatierung (Track 35-41, einzelne Tracks, doppelte Tracks, Header-Parameter ändern, zerstörte Sektoren, gleiche Sektoren auf einem Track)
- Halftracks, der Track mit den Nullen, kompletten Track prüfen
- Alle Tricks der SYNC-Markierungen (Killertracks, überlange SYNC-Markierungen, 3000-SYNC-Schutz, Daten ohne SYNC-Markierungen)
- Der Disketten-Killer (nur Ausschalten der Floppy hilft)
- Alle Tricks der Knack-Module und wie man sich dagegen schützt
- Was Software-Häuser über Cracker wissen sollten

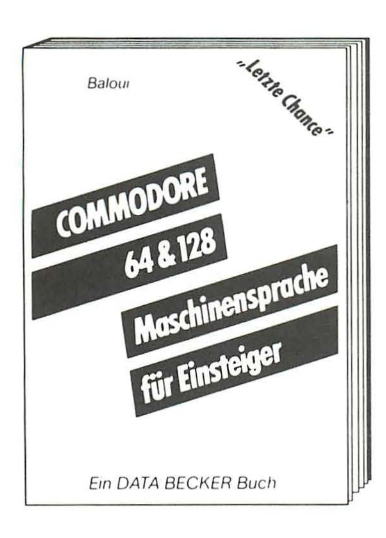
Gelfand, Felt, Strauch, Krsnik
Das Anti-Cracker-Buch
384 Seiten, DM 39,–
ISBN 3-89011-253-6

Bücher zum Commodore 64



Bücher zum Commodore 64

Endlich ist es soweit: Maschinensprache für Einsteiger ist das Buch, auf das alle, die sich schon immer für Maschinensprache interessierten, gewartet haben. Finden Sie heraus, was in Ihnen und in Ihrem Rechner steckt. Dieses Buch zeigt Ihnen, wie's geht: ohne Fachchinesisch, dafür einfach, schnell und effektiv. Nutzen Sie diese Chance.



Aus dem Inhalt:

- Einführung in Assembler – Was man über Zahlen, Speicher und den Rechner wissen sollte
- Wie man mit einem Monitor arbeitet (Laden, Starten, Eingeben und Speichern eigener Programme)
- Die ersten Befehle und die Adressierungsarten
- Fortgeschrittene Assemblerprogrammierung
- Noch mehr über Schleifen
- Rechnen in Maschinensprache
- Vergleichsbefehle
- Stapeloperationen
- Interruptprogrammierung
- Die verschiedenen Rechner/Programmierhilfsmittel für C16, 116, Plus/4 und C128
- Und viele, viele Beispiele . . .

Baloui
Maschinensprache für Einsteiger
346 Seiten, DM 29,–
ISBN 3-89011-182-3

DAS STEHT DRIN:

Tips & Tricks braucht jeder. Aber wer will schon lange suchen und blättern. In diesem Buch sind alle Informationen nach Anwendungsgebieten geordnet. Egal, ob Sie effektiven Sound, Grafik oder BASIC programmieren wollen. Hier finden Sie für jede Anwendung den richtigen Tip.

Aus dem Inhalt:

- Peeks & Pokes
- Einzeiler
- Floppyzugriffszeit beschleunigen
- Disketten versiegeln
- Unscratch
- Direktory-Kill
- BASIC beschleunigen
- ReNew
- Zurücksetzen von Arrays
- 16 Funktionstasten
- Windows in BASIC
- Spiele-Hilfen
- Spiele-Pokes
- GEOS-Anwendungen

UND GESCHRIEBEN HAT DIESES BUCH:

Andreas Polk, der in der langen Zeit als begeisterter C64-Anwender und Hobbyprogrammierer seine Erfahrungen zusammengetragen hat, schaffte mit diesem Titel den Einstieg als Autor.


ISB N 3-89011-281-1 DM +029.00

DM 29,–
ÖS 226,–
sFr 27,–

02900

DATA BECKER

9 783890 112817